

# Package ‘multiplex’

September 21, 2018

**Type** Package

**Version** 2.9

**Depends** R (>= 3.4.0)

**Imports** methods

**Suggests** multigraph, Rgraphviz, knitr

**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. Among other things, it makes possible to create and manipulate multivariate network data with different formats, and there are effective ways available to treat multiple networks with routines that combine algebraic systems like the partially ordered semigroup or the semiring structure together with the relational bundles occurring in different types of multivariate network data sets. It also provides an algebraic approach for two-mode networks through Galois derivations between families of the pairs of subsets in the two domains.

**Date** 2018-09-21

**Author** Antonio Rivero Ostoic [aut, cre]

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

**URL** <http://github.com/mplex/multiplex/>

**BugReports** <https://github.com/mplex/multiplex/issues/>

**Repository** CRAN

**Encoding** latin1

**License** GPL-3

**VignetteBuilder** knitr

**NeedsCompilation** no

**Date/Publication** 2018-09-21 12:00:03 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 . . . . .	14
diagram . . . . .	15
diagram.levels . . . . .	17
dichot . . . . .	18
edgeT . . . . .	19
expos . . . . .	20
fact . . . . .	22
fltr . . . . .	23
galois . . . . .	24
hierar . . . . .	26
incubs . . . . .	27
isom . . . . .	28
ltlw . . . . .	29
mnplx . . . . .	30
neighb . . . . .	31
pacnet . . . . .	32
partial.order . . . . .	34
perm . . . . .	35
pi.rels . . . . .	36
prev . . . . .	37
rbox . . . . .	38
read.dl . . . . .	39
read.gml . . . . .	40
read.srt . . . . .	41
reduc . . . . .	43
rel.sys . . . . .	44
rm.isol . . . . .	46
semigroup . . . . .	47
semiring . . . . .	49
signed . . . . .	50
strings . . . . .	52
summaryBundles . . . . .	53
transf . . . . .	54
wordT . . . . .	56
write.dat . . . . .	57
write.dl . . . . .	58
write.gml . . . . .	59
write.srt . . . . .	60
zbind . . . . .	61

## Description

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

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

## Details

Package:	multiplex
Type:	Package
Version:	2.9
Date:	21 September 2018
License:	GPL-3
LazyLoad:	yes

To work with this package 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. This algebraic structure is constructed by the `semigroup` function, 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 completely the semigroup, and they are constructed with the functions `wordT` and `edgeT` respectively. Unique relations of the complete semigroup are given by the `strings` function. The `partial.order` function specifies the ordering of the string elements in the semigroup. The function `diagram` produces the lattice of inclusions of a structure having ordered relations.

Semigroups can be analysed further by `l1lw` function, and they also can be reduced by a decomposition process. The decomposition is based on congruence or  $\pi$ -relations of the unique strings imported from Pacnet. In this case `pi.rels`, `cngr`, and `decomp` will make this job for you either for an abstract or a partially ordered structure.

In addition, it is possible to analyse structural balance in signed networks, which are built by signed, through the algebraic structure of the semiring. A semiring is an algebraic structure that combines an abstract semigroup with identity under multiplication and a commutative monoid under addition. The `semiring` function is capable to perform both balance and cluster semiring either with cycles or semicycles.

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

The ‘multiplex’ package creates a Relation-Box with the `rbox` function, and it implements the Partial Structural Equivalence expressed in the cumulated person hierarchy of the network calculated via the `cph` function.

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)ceive (t)ies format that can be loaded and transformed via the `read.srt` function. Other formats for multiple network data like Ucinet d1 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. ‘Creating context for social influence processes in multiplex networks.’ *Network Science*, 5(1), 1-29.

### See Also

[multigraph](#), [bmgraph](#)

### 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**

<code>x</code>	a matrix representing the signed network
<code>lbs</code>	(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**

<code>x</code>	an array; usually with three dimensions of stacked matrices where the multiple relations are placed.
<code>lbs</code>	(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 a given multiple network.

## 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 the multiple network. A bundle is a special 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, which are included 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. If loops were considered, these are at the end of the table.

More detailed information about the bundle class occurrences is given by the function [bundles](#).

## Note

Neither loops or null dyads are regarded as properly bundle classes.

## 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, simpl = 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
simpl	(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 special type of pattern made of relations at different levels that is binding a pair of nodes or actors in a network of relations. A bundle class is a dyadic configuration resulting from the mixture of the direction and the types of tie 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 the detailed information about the bundle class patterns in the multiple network 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 simpl, whereas the collapse option blurs the levels in the strings. The input array is always dichotomized.

**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**

It is also possible to obtain the total number of occurrences in each bundle class pattern by 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**

Congruences 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
congr(S, PO=NULL)
```

---

comps

*Find components in multiple networks*

---

**Description**

Function to find different components in the network plus 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 by means of transitive closure of the bundle ties. By default the 'entire' network is chosen, but the option bonds allows to discriminate different types of relational bundles.

**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 and isolates
comps(arr)
```

---

cph

*Cumulated Person Hierarchy*

---

**Description**

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

**Usage**

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

**Arguments**

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

**Details**

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

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

**Value**

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

**Note**

If the length of the labels differ 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("mc", "pi", "at", "cc"), reduc, fac)
```

**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 congruence class (option "cc") or rather on a $\pi$ -relation, atoms, or a meet-complement 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

**Details**

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

**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 the Hasse Diagram of a set of ordered relations*


---

**Description**

A function to plot the 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, ...)
```

**Arguments**

<code>x</code>	a matrix representing ordered relations
<code>attrs</code>	(optional) attributes of the diagram
<code>main</code>	(optional) title of the diagram
<code>incmp</code>	(optional and logical) whether or not the incomparable elements should be included in the lattice diagram
<code>cex.main</code>	(optional) size of the diagram's title
<code>bg</code>	(optional) the background color of the diagram
<code>mar</code>	(optional) the margins of the plot
<code>shape</code>	(optional) the shape of the vertices
<code>col</code>	(optional) the color of the vertices
<code>col0</code>	(optional) the color of the vertices' contour
<code>fc0l</code>	(optional) the color of the text's vertices
<code>ecol</code>	(optional) the color of the edges
<code>lty</code>	(optional) the shape of the edges
<code>lbs</code>	(optional) labels of the elements in the partially ordered set
<code>ffamily</code>	(optional) the font family of the vertex labels
<code>fstyle</code>	(optional) the font style of the vertex labels with options: 'bold', 'italic', 'bolditalic'
<code>fsize</code>	(optional) the font size of the vertex labels
<code>...</code>	(optional) additional graphical items

**Details**

An example of ordered relations is found in the partial order table of relations, which is 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.

**Value**

A Hasse diagram of the partial order relation.

**Warning**

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

**Note**

Note that if the elements of the partial order are not labelled, Roman numerals will be given to each element.



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

**Usage**

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

**Arguments**

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

**Details**

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

**Value**

A 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*

---

**Description**

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

**Usage**

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

**Arguments**

<code>x</code>	some data in a numeric form (usually arrays)
<code>c</code>	the cutoff value to perform the dichotomization (default 1)
<code>diag</code>	(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 system can be the entire network or configuration with strong or weak bonds among the actors. It is possible to specify different behaviours 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 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. <a href="#">rel.sys</a> )
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

*Factorisation of semigroup structures*


---

### Description

A function to decompose partially ordered semigroups

### Usage

```
fact(S, P, uniq = TRUE, fac, atoms, mc, atm, patm, k)
```

### Arguments

S	The semigroup object
P	The partial order structure associada to S
uniq	(logical) whether the factorisation include the unique induced inclusions
fac	the ‘factor’ to be factorised, in case that input factorised partially ordered structures
atoms	(logical) whether or not include the atoms in the output
mc	(logical) whether or not include the meet-complements in the output
atm	(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 factorisation is part of the decomposition process for partially ordered semigroups.

### 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
mc	(optional) meet-complements
note	(optional) induced inclusions without the substitution property

**Author(s)**

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

**References**

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

**See Also**

[decomp](#), [cngr](#), [pacnet](#)

**Examples**

```
## Create a partially ordered semigroup
arr <- round( replace( array(runif(18), c(3,3,2)), array(runif(18),
  c(3,3,2))>.5, 1 ) )
# semigroup
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, rclos = TRUE, ideal = FALSE)
```

**Arguments**

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

**Details**

This function helps to find principal filters or principal ideals for an element in a partial order structure. Such inputs are typically 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 an object or an attribute by a character name, which should be part in the labels of the dimensions of the partial order table with a reduced labeling. Principal filters with a full labelling are not allowed if the reference element is an object or an attributes. 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, rclose=TRUE)
```

---

galois

*Galois derivations between subsets*


---

**Description**

Function to perform Galois derivations between partially ordered subsets

**Usage**

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



**Arguments**

x	a data frame with objects and attributes
labeling	whether the derivations should be with full or reduced labeling

**Details**

Galois derivations (or connections) are mappings between families of partially ordered subsets of elements. Such derivations are useful to analyse the structure of both subsets, which in a social network are typically the actors and their corresponding affiliations or events, i.e. two-mode networks, or else a group of objects with a list of different attributes 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 cumulation of the 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

*Business Network Incubators Data Sets*


---

**Description**

These are four data sets collected in year 2010 (see ‘source’ for the details) of multiple relations between entrepreneurial firms working in business incubators in Denmark.

Each data set contains the adjacency matrices of the three social relations, coded as C, F, and K for working collaboration, informal friendship, and perceived competition among the firms. There are also two actor attributes corresponding to the adoption of a Web innovations at that time 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 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, etc.

**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 dimensionanl 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 putted in together in `incubs`.

In order 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` citation and useful information

### Source

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

---

isom

*Make Isomorphic...*

---

### Description

Make isomorphic a vector or a data frame object.

### Usage

```
isom(x, uniq = FALSE)
```

### Arguments

`x` a vector or a data frame

`uniq` (logical) whether or not the unique class should be returned

### Details

This function serves to normalize otherwise arbitrary numbering of a vector or a data frame object data into a progressive ordinal recount of the data in the specified object. This is especially useful to specify more properly the corresponding clustering vector of a set of elements in a given system.

### Value

The input data with isomorphic classes.

In case that `uniq` is specified, then a list with the following items is given:

`ism` the input data with isomorphic classes

`uniq` the unique isomorphic classes

**Author(s)**

Antonio Rivero Ostoic

**See Also**[cngr](#), [reduc](#)**Examples**

```
## Create a data frame (from 'Vectors')
df <- data.frame(x=1:3, y=5:7)

## Make isomorphic the observations
isom(df)
```

ltlw

*First- and Last Letter Laws***Description**

Find the First- and Last Letter Laws in the semigroup of relations.

**Usage**

```
ltlw(x)
```

**Arguments**

`x` an object from the ‘Semigroup’ class.

**Details**

The First- and Last Letter Laws of the semigroup of relations correspond to the right and left zero of the semigroup, and it has some consequences in the interpretation of the results of the relational structure of the network. This function depends on the semigroup of a (recommended) labeled set of relationships.

**Value**

<code>S</code>	the semigroup of relations.
<code>strings</code>	the strings of relations in the semigroup.
<code>First.Letter</code>	the First.Letters in rows and columns
<code>Last.Letter</code>	the Last.Letters in rows and columns

**Note**

Since this function is based on the semigroup construction, see the Warning section and look at other details in the [semigroup](#) function section.

**Author(s)**

Antonio Rivero Ostoic

**References**

Lorrain, F. *Réseaux sociaux et classifications sociales*. Hermann, Paris. 1975.

**See Also**

[semigroup](#), [strings](#).

**Examples**

```
## Create the data: 3 binary relations among 3 elements
arr <- round( replace( array(runif(27), c(3,3,3)), array(runif(27),
  c(3,3,3))>.75, 1 ) )

## Put labels
dimnames(arr)[[3]] <- list("n", "m", "ñ")

## Construct the semigroup
S <- semigroup(arr, type="symbolic")

## Find the Letter Laws
l1lw(S)
```

---

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
mnp1x(arr)
```

---

neighb

*Neighborhood of an actor or group of actors*


---

**Description**

A function to find the neighborhood 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 network is <i>undirected</i> (default) und; directed with <i>incoming node's</i> ties inn to the the reference actor, or else with <i>outgoing</i> arcs out
k	the “distance” of the neighbor 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.

**Value**

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

**Note**

In case that the reference actors are in different components of the network, the output does not discriminate this fact.

**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 immediate 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)
```

### Arguments

x	an object of a 'Strings' or a 'Galois' class
type	whether the object corresponds to string relations, Galois derivations, or $\pi$ -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 $\pi$ -relations be included

### 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)
```

**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 should be reverted.
lbs	(optional) the labels after the permutation

**Details**

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

**Value**

A permuted matrix or array

**Author(s)**

Antonio Rivero Ostoic

**See Also**

[cph](#), [partial.order](#)

**Examples**

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

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

---

pi.rels

 $\pi$ -Relations

---

**Description**

A function to establish the  $\pi$ -relations of a partially ordered structure coming 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 $\pi$ -relations, eventually with the partial order
mc	the meet-complements

**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 in order to get the partial right multiplication table.

That is because the amount of undefined data in such 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 relative large number of elements for a semigroup construction of 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 relative high. The semigroup construction can however still take long time without such message; cf. 'Details' for this.

**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, simpl = FALSE, k = 3, tlbs)
```

**Arguments**

<code>w</code>	an array with three dimensions of stacked matrices of generating relations.
<code>transp</code>	(logical) whether or not the transpose of each matrix in <code>w</code> should be included.
<code>simpl</code>	(logical) whether to simplify or not the strings of relations
<code>k</code>	length of the Relation-Box in $z$
<code>tlbs</code>	(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. In order 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.

<code>w</code>	the primitive relations in the Relation-Box
<code>W</code>	the structure of the Relation-Box
<code>lbs</code>	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 is supported. With many types of relations, and when the order of the multiple 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
rbox(incubA$IM)
```

---

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 both to the nodes and the edges for visualization purposes.

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

**Value**

Depending the option chosen, the output is either a data frame or an array representing the multi-graph. 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 srt Files*

---

**Description**

A function to read files with *send*, *receive*, and *ties* format for a multivariate network with the possibility to transform it into an 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 3 columns for the sender, receiver, and the ties, one column for each type of relation. However, the `attr` option correspond to a actor and self-ties data frame file with the option to transform it into a diagonal matrix. When `toarray` is set to `FALSE`, options `attr` and `rownames` allow placing the first column of the data frame as the name of the table, which is the format of two-mode data, and compute for instance Galois transformations among the partite sets. If more than one isolate is added, then the data must be included as a vector.

It is also possible to treat the input data as data frame object and manipulate it 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

Valued networks are now supported as well.

## Author(s)

Antonio Rivero Ostoic

## See Also

[write.srt](#), [read.gml](#), [read.dl](#), [galois](#)

---

reduc	<i>Reduce a matrix or array</i>
-------	---------------------------------

---

**Description**

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

**Usage**

```
reduc(x, clu, lbs = NULL)
```

**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

**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 a 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 multiple 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 pattern exclusively without mutual character.

When selecting from a list with actor attributes, it is also possible to select the network members having or *not* the attribute that is specified in the *Attrs* output. Use *att* or *noatt* for the two options.

**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 isolate nodes in simple and multiple networks.

**Usage**

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

**Arguments**

<code>x</code>	a matrix or array representing a network
<code>diag</code>	(optional and logical) if arrays, should the diagonals be included in the computation?
<code>diag.incl</code>	(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 is no edges adjacent to these kinds of nodes at any level.

**Value**

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

**Author(s)**

Antonio Rivero Ostoic

**See Also**

[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)
```

**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"), lbs = NULL,
          cmp = FALSE, smp1 = FALSE)
```

**Arguments**

x	an array; usually with three dimensions of stacked matrices where the multiple relations are placed.
type	does the semigroup should be given in a numerical (default) or in a symbolic form?
lbs	(optional) a list of the labels of each distinct relation.
cmp	(optional) a logical to indicate whether the composite matrices should be also given in the output.
smp1	(logical) whether to simplify or not the strings of relations.

**Details**

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

The semigroup, which is an algebraic structure having a set with an associative operation on it, is calculated considering binary matrices only. This means that if the provided matrices are valued, the function will dichotomise the input data automatically; values higher or equal to a unit are converted to one, otherwise they are set to zero. If you are not happy with that, you can go to [dichot](#) and specify your own cutoff value for the dichotomization.

**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 is returned as a data frame with the strings of the semigroup.

### Warning

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

### Note

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

### Author(s)

Antonio Rivero Ostoic

### References

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

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

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

### See Also

[prev](#), [strings](#), [edgeT](#), [wordT](#), [dichot](#), [ltrlw](#), [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 determinate 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 coerce a array(s) to become a ‘Signed’ object. Positive ties are always in the first argument, and in case that this array has three dimensions, then the second dimension is considered as the negative ties, provided that N is 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, simpl)
```

**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
simpl	(optional and logical) whether to simplify or not the string relations

**Details**

The strings are the unique relations which constitutes 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](#), [ltnw](#).

## 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	<i>Summary of Bundle Classes</i>
----------------	----------------------------------

---

## 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 a 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 `file` path is activated to obtain a tex file with the different bundle class patterns. Finally, the optional argument `byties` provides more precise information of the patterned ties disregarding the relational content.

**Value**

The distinct bundle class patterns with a user friendly 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**

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"), lbs = NULL, lb2lb,
sep, ord, sort, add, adc)
```

**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, or from a matrix to a list of pair relations, or else from a list of pair relations to a square array
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?
add	(optional) add elements in the array's 'domain'
adc	(optional) add elements in the array's 'codomain'

**Details**

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

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

**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, the collection of unique relations (both compound an generators) are represented by nodes, and the generators are edges that record the result of post-multiplying the compound relations by the generators (Pattison, 1993).

**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 neither in the “node” nor the “generator” of the Word table since they are not product of any other element in the semigroup. cf. details for the rest of the values.

**Note**

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

**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](#).

## 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 both to 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	<i>Write srt Files</i>
-----------	------------------------

---

## 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 send, receive, and ties, 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	<i>Combine Arrays</i>
-------	-----------------------

---

**Description**

Combine multidimensional arrays.

**Usage**

```
zbind(...)
```

**Arguments**

...                    One or more arrays with two or three dimensions

**Details**

This function is for stacking 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. The dimnames in the output correspond to the first array in the input, and a Warning message is given when these are NULL.

**Value**

Usually a three dimensional array

**Note**

Data frames should be transformed into arrays

**Author(s)**

Antonio Rivero Ostoic

**See Also**

[mnp1x](#), [dichot](#), [strings](#)

**Examples**

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

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

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

# Index

## \*Topic **IO**

- multiplex-package, 3
- pacnet, 32
- read.dl, 39
- read.gml, 40
- read.srt, 41
- summaryBundles, 53
- write.dat, 57
- write.dl, 58
- write.gml, 59
- write.srt, 60

## \*Topic **algebra**

- as.semigroup, 5
- as.strings, 7
- cngr, 11
- cph, 13
- decomp, 14
- diagram.levels, 17
- edgeT, 19
- fact, 22
- fltr, 23
- galois, 24
- hierar, 26
- ltlw, 29
- partial.order, 34
- perm, 35
- pi.rels, 36
- prev, 37
- rbox, 38
- semigroup, 47
- semiring, 49
- strings, 52
- wordT, 56

## \*Topic **array**

- as.signed, 6
- as.strings, 7
- cph, 13
- decomp, 14
- mnplx, 30

- perm, 35
- rbox, 38
- reduc, 43
- rm.isol, 46
- signed, 50
- strings, 52
- transf, 54
- zbind, 61

## \*Topic **attribute**

- bundles, 9
- expos, 20
- multiplex-package, 3
- rel.sys, 44

## \*Topic **cluster**

- cngr, 11
- comps, 12
- decomp, 14
- perm, 35
- reduc, 43

## \*Topic **datasets**

- incubs, 27

## \*Topic **data**

- bundle.census, 8
- bundles, 9
- multiplex-package, 3
- pacnet, 32
- read.dl, 39
- read.gml, 40
- read.srt, 41
- write.dat, 57
- write.dl, 58
- write.gml, 59
- write.srt, 60

## \*Topic **file**

- multiplex-package, 3
- pacnet, 32
- read.dl, 39
- read.gml, 40
- write.dat, 57

- write.dl, 58
  - write.gml, 59
  - write.srt, 60
- \*Topic **graphs**
  - diagram, 15
- \*Topic **list**
  - bundles, 9
- \*Topic **manip**
  - diagram.levels, 17
  - dichot, 18
  - isom, 28
  - mnplx, 30
  - multiplex-package, 3
  - neighb, 31
  - perm, 35
  - read.srt, 41
  - reduc, 43
  - rm.isol, 46
  - transf, 54
  - zbind, 61
- \*Topic **math**
  - as.semigroup, 5
  - bundles, 9
  - cngr, 11
  - cph, 13
  - decomp, 14
  - fact, 22
  - fltr, 23
  - galois, 24
  - hierar, 26
  - ltlw, 29
  - partial.order, 34
  - perm, 35
  - prev, 37
  - semigroup, 47
  - semiring, 49
- \*Topic **models**
  - as.signed, 6
  - comps, 12
  - expos, 20
  - multiplex-package, 3
  - neighb, 31
  - rel.sys, 44
  - semiring, 49
  - signed, 50
- \*Topic **print**
  - summaryBundles, 53
- as.semigroup, 5
- as.signed, 6, 50, 51
- as.strings, 7, 17, 34
- bmgraph, 4
- bundle.census, 8, 10, 54
- bundles, 8, 9, 9, 12, 21, 32, 45, 53–55
- cngr, 11, 15, 23, 29, 33, 43, 48
- comps, 12
- cph, 13, 26, 27, 35, 39
- decomp, 11, 14, 23, 33, 37, 43
- diagram, 14, 15, 17, 18, 24–26, 34
- diagram.levels, 17, 17
- dichot, 18, 31, 47, 48, 61
- edgeT, 19, 38, 48, 57
- expos, 20, 32, 45
- fact, 11, 15, 22
- fltr, 23, 25, 34
- galois, 17, 24, 24, 34, 42
- hierar, 26, 39
- incA (incubs), 27
- incB (incubs), 27
- incC (incubs), 27
- incD (incubs), 27
- incubA (incubs), 27
- incubB (incubs), 27
- incubC (incubs), 27
- incubD (incubs), 27
- incubs, 27
- isom, 28
- ltlw, 29, 48, 53
- mnplx, 30, 61
- multigraph, 4, 28
- multiplex-package, 3
- neighb, 21, 31, 45
- pacnet, 11, 23, 32, 37, 58
- partial.order, 7, 15–18, 24–26, 34, 35, 53
- perm, 18, 34, 35
- pi.rels, 15, 33, 36
- prev, 19, 37, 48

rbox, *14, 26, 38, 43*  
read.dl, *39, 41, 42, 59*  
read.gml, *40, 40, 42, 59*  
read.srt, *40, 41, 41, 46, 55, 60*  
reduc, *15, 29, 31, 43, 55*  
rel.sys, *12, 21, 32, 44, 55*  
replace, *19*  
rm.isol, *46*

semigroup, *5, 6, 14, 15, 19, 20, 29, 30, 34, 38,*  
*39, 47, 53, 57*  
semiring, *6, 49, 51*  
signed, *6, 50, 50*  
strings, *7, 17, 30, 34, 48, 52, 56, 61*  
subset, *42*  
summaryBundles, *9, 10, 53*

transf, *10, 54*

wordT, *20, 48, 56*  
write.dat, *33, 57, 59*  
write.dl, *40, 58, 58, 59, 60*  
write.gml, *41, 58, 59, 59*  
write.srt, *42, 59, 60*

zbind, *7, 31, 46, 61*