

# Package ‘blockmodeling’

January 2, 2012

**Type** Package

**Title** An R package for Generalized and classical blockmodeling of valued networks

**Version** 0.1.8

**Date** 2010-01-12

**Imports** stats

**Suggests** sna, Matrix

**Author** Ales Ziberna

**Maintainer** Ales Ziberna <ales.ziberna@gmail.com>

**Description** The package is primarily ment as an implementation of Generalized blockmodeling for valued networks. In addition,measures of similarity or dissimilarity based on structural equivalence and regular equivalence (REGE algorithm) can be computed and partitioned matrices can be plotted.

**License** GPL (>= 2)

**Encoding** UTF-8

**Repository** CRAN

**Repository/R-Forge/Project** blockmodeling

**Repository/R-Forge/Revision** 26

**Date/Publication** 2010-01-13 15:43:51

## R topics documented:

blockmodeling-package . . . . .	2
check.these.par . . . . .	4
clu . . . . .	6
crit.fun . . . . .	7
find.m . . . . .	12

formatA . . . . .	14
fun.by.blocks . . . . .	15
genRandomPar . . . . .	16
gplot1 . . . . .	18
ircNorm . . . . .	19
nkpartitions . . . . .	20
opt.par . . . . .	21
opt.random.par . . . . .	24
Pajek . . . . .	28
plot.mat . . . . .	30
rand . . . . .	35
recode . . . . .	36
REGE . . . . .	36
reorderImage . . . . .	39
sedist . . . . .	40
ss . . . . .	41
two2one . . . . .	42

## Index 44

blockmodeling-package *An R package for Generalized and classical blockmodeling of valued networks*

### Description

This package is primarily meant as an implementation of Generalized blockmodeling. In addition, functions for computation of (dis)similarities in terms of structural and regular equivalence, plotting and other "utility" functions are provided.

### Author(s)

Aleš Žiberna

### References

- ŽIBERNA, Aleš (2006): Generalized Blockmodeling of Valued Networks. *Social Networks*, Jan. 2007, vol. 29, no. 1, 105-126. <http://dx.doi.org/10.1016/j.socnet.2006.04.002>.
- ŽIBERNA, Aleš. Direct and indirect approaches to blockmodeling of valued networks in terms of regular equivalence. *J. math. sociol.*, 2008, vol. 32, no. 1, 57-84. <http://www.informaworld.com/smpp/content?content=10.1080/00222500701790207>.
- DOREIAN, Patrick, BATAGELJ, Vladimir, FERLIGOJ, Anuška (2005): Generalized blockmodeling. (Structural analysis in the social sciences, 25). Cambridge [etc.]: Cambridge University Press, 2005. XV, 384 p., ISBN 0-521-84085-6.
- White, D. R., K. P. Reitz (1983): "Graph and semigroup homomorphisms on networks of relations". *Social Networks*, 5, p. 193-234.
- White, Douglas R.(2005): REGGE (web page). <http://eclectic.ss.uci.edu/~drwhite/REGGE/> (12.5.2005).

**See Also**

Packages: [sna network](#)

Functions inside this package: [crit.fun](#), [opt.par](#), [opt.random.par](#), [opt.these.par](#), [check.these.par](#), [REGE](#), [plot.mat](#)

**Examples**

```
n<-8 #if larger, the number of partitions increases dramatically,
# as does if we increase the number of clusters
net<-matrix(NA,ncol=n,nrow=n)
clu<-rep(1:2,times=c(3,5))
tclu<-table(clu)
net[clu==1,clu==1]<-rnorm(n=tclu[1]*tclu[1],mean=0,sd=1)
net[clu==1,clu==2]<-rnorm(n=tclu[1]*tclu[2],mean=4,sd=1)
net[clu==2,clu==1]<-rnorm(n=tclu[2]*tclu[1],mean=0,sd=1)
net[clu==2,clu==2]<-rnorm(n=tclu[2]*tclu[2],mean=0,sd=1)

#we select a random partition and then optimise it

all.par<-nkpartitions(n=n, k=length(tclu))
#forming the partitions
all.par<-lapply(apply(all.par,1,list),function(x)x[[1]])
# to make a list out of the matrix

#optimizing one partition
res<-opt.par(M=net,
  clu=all.par[[sample(1:length(all.par),size=1)]],
  approach="ss", blocks="com")
plot(res) #Hopefully we get the original partition

#optimizing 10 random partitions which with opt.these.par
res<-opt.these.par(M=net,
  partitions=all.par[sample(1:length(all.par),size=10)],
  approach="ss",blocks="com")
plot(res) #Hopefully we get the original partition

#optimizing 10 random partitions with opt.random.par
res<-opt.random.par(M=net,k=2,rep=10,approach="ss",blocks="com")
plot(res) #Hopefully we get the original partition

#Checking all possible partitions
nkpar(n=n, k=length(tclu)) #computing the number of partitions
all.par<-nkpartitions(n=n, k=length(tclu))
#forming the partitions
all.par<-lapply(apply(all.par,1,list),function(x)x[[1]])
# to make a list out of the matrix
res<-check.these.par(M=net,partitions=all.par,approach="ss",
  blocks="com")
plot(res) #we get the original partition

#using indidect approach - structural equivalence
D<-sedist(M=net)
```

```
plot.mat(net, clu=cutree(hclust(d=D,method="ward"),k=2))
```

---

check.these.par	<i>Computes the value of a criterion function for a given network and a set of partitions</i>
-----------------	---

---

### Description

The function computes the value of a criterion function for a given network and a set of partitions for Generalized blockmodeling. (Žiberna, 2006) based on other parameters (see below and [crit.fun](#)).

### Usage

```
check.these.par(M, partitions, approach, return.err = TRUE,
  save.initial.param = TRUE, force.fun = NULL, ...)
```

### Arguments

M	A matrix representing the (usually valued) network. For now, only one-relational networks are supported. The network can have one or more modes (different kinds of units with no ties among themselves). If the network is not two-mode, the matrix must be square.
partitions	A list of partitions. Each unique value represents one cluster. If the network is one-mode, then this should be a vector, else a list of vectors, one for each mode.
approach	One of the approaches described in Žiberna (2006). Possible values are: "bin" - binary blockmodeling, "val" - valued blockmodeling, "imp" - implicit blockmodeling, "ss" - sum of squares homogeneity blockmodeling, and "ad" - absolute deviations homogeneity blockmodeling.
return.err	Should the error for each evaluated partition be returned
save.initial.param	Should the initial parameters (approach,...)
force.fun	Select the function used to evaluate the network and a partition. This should be used only in extreme cases. Otherwise, the appropriate function is selected (generated) based on the input parameters.
...	Arguments to <code>gen.crit.fun</code> see <a href="#">crit.fun</a> for description. Some are required!!!

### Value

M	The matrix of the network analyzed
best	A list of results from <code>crit.fun.tmp</code> with the same elements as the result of <code>crit.fun</code> , only without M
err	If selected - The vector of errors or inconsistencies of the empirical network with the ideal network for a given blockmodel (model,approach,...) and partitions

call            The call used to call the function.  
 initial.param   If selected - The initial parameters used.  
 ...

### Warning

This function is usually used to check all possible partitions. If the number of partitions is large (several 1000), this can be extremely time demanding. It is advisable to first time the function on a smaller subset.

### Author(s)

Aleš Žiberna

### References

ŽIBERNA, Aleš (2006): Generalized Blockmodeling of Valued Networks. *Social Networks*, Jan. 2007, vol. 29, no. 1, 105-126. <http://dx.doi.org/10.1016/j.socnet.2006.04.002>.

ŽIBERNA, Aleš. Direct and indirect approaches to blockmodeling of valued networks in terms of regular equivalence. *J. math. sociol.*, 2008, vol. 32, no. 1, 57-84. <http://www.informaworld.com/smp/content?content=10.1080/00222500701790207>.

DOREIAN, Patrick, BATAGELJ, Vladimir, FERLIGOJ, Anuška (2005): Generalized blockmodeling. (Structural analysis in the social sciences, 25). Cambridge [etc.]: Cambridge University Press, 2005. XV, 384 p., ISBN 0-521-84085-6.

### See Also

[crit.fun](#), [opt.par](#), [opt.these.par](#), [nkpartitions](#), [plot.check.these.par](#)

### Examples

```
n<-8 # if larger, the number of partitions increases dramatically,
      # as does if we increase the number of clusters
net<-matrix(NA,ncol=n,nrow=n)
clu<-rep(1:2,times=c(3,5))
tclu<-table(clu)
net[clu==1,clu==1]<-rnorm(n=tclu[1]*tclu[1],mean=0,sd=1)
net[clu==1,clu==2]<-rnorm(n=tclu[1]*tclu[2],mean=4,sd=1)
net[clu==2,clu==1]<-rnorm(n=tclu[2]*tclu[1],mean=0,sd=1)
net[clu==2,clu==2]<-rnorm(n=tclu[2]*tclu[2],mean=0,sd=1)

#computation of criterion function with the correct partition
nkpar(n=n, k=length(tclu)) #computing the number of partitions
all.par<-nkpartitions(n=n, k=length(tclu))
#forming the partitions
all.par<-lapply(apply(all.par,1,list),function(x)x[[1]])
# to make a list out of the matrix
res<-check.these.par(M=net,partitions=all.par,approach="ss",
  blocks="com")
plot(res) #we get the original partition
```

---

clu *Function for extraction of some elements for objects, returned by functions for Generalized blockmodeling*

---

### Description

Function for extraction of clu (partition), all best clus (partitions), IM (image or blockmodel) and err (total error or inconsistency) for objects, returned by functions [opt.par](#), [opt.random.par](#), [opt.these.par](#), and [check.these.par](#)

### Usage

```
clu(res, which = 1, ...)
IM(res, which = 1, ...)
err(res, ...)
partitions(res)
```

### Arguments

res	Result of function <a href="#">opt.par</a> , <a href="#">opt.random.par</a> , <a href="#">opt.these.par</a> , or <a href="#">check.these.par</a>
which	From which (if there are more than one) "best" solution should the element be extracted. Warning! which greater than the number of "best" partitions produces an error.
...	Not used

### Value

The desired element.

### Author(s)

Aleš Žiberna

### References

ŽIBERNA, Aleš (2006): Generalized Blockmodeling of Valued Networks. *Social Networks*, Jan. 2007, vol. 29, no. 1, 105-126. <http://dx.doi.org/10.1016/j.socnet.2006.04.002>.

ŽIBERNA, Aleš. Direct and indirect approaches to blockmodeling of valued networks in terms of regular equivalence. *J. math. sociol.*, 2008, vol. 32, no. 1, 57-84. <http://www.informaworld.com/smpp/content?content=10.1080/00222500701790207>.

DOREIAN, Patrick, BATAGELJ, Vladimir, FERLIGOJ, Anuška (2005): Generalized blockmodeling. (*Structural analysis in the social sciences*, 25). Cambridge [etc.]: Cambridge University Press, 2005. XV, 384 p., ISBN 0-521-84085-6.

### See Also

[crit.fun](#), [check.these.par](#), [opt.random.par](#), [opt.these.par](#), [plot.opt.par](#)

**Examples**

```

n<-8 #if larger, the number of partitions increases dramatically,
      #as does if we increase the number of clusters
net<-matrix(NA,ncol=n,nrow=n)
clu<-rep(1:2,times=c(3,5))
tclu<-table(clu)
net[clu==1,clu==1]<-rnorm(n=tclu[1]*tclu[1],mean=0,sd=1)
net[clu==1,clu==2]<-rnorm(n=tclu[1]*tclu[2],mean=4,sd=1)
net[clu==2,clu==1]<-rnorm(n=tclu[2]*tclu[1],mean=0,sd=1)
net[clu==2,clu==2]<-rnorm(n=tclu[2]*tclu[2],mean=0,sd=1)

#we select a random partition and then optimise it

all.par<-nkpartitions(n=n, k=length(tclu))
#forming the partitions
all.par<-lapply(apply(all.par,1,list),function(x)x[[1]])
# to make a list out of the matrix
res<-opt.par(M=net,
             clu=all.par[[sample(1:length(all.par),size=1)]],
             approach="ss",blocks="com")
plot(res) #Hopefully we get the original partition
clu(res) #Hopefully we get the original partition
err(res) #Error
IM(res) #NULL, because FORTRAN subroutine is used.

```

crit.fun

*Computes the criterion function for a given network and partition***Description**

The function computes the value of a criterion function for a given network and partition for Generalized blockmodeling. (Žibera, 2006) based on other parameters (see below).

**Usage**

```
crit.fun(M, clu, approach, ...)
```

**Arguments**

M	A matrix representing the (usually valued) network. For multi-relational networks, this should be an array with the third dimension representing the relation. The network can have one or more modes (different kinds of units with no ties among themselves. If the network is not two-mode, the matrix must be square.
clu	A partition. Each unique value represents one cluster. If the network is one-mode, than this should be a vector, else a list of vectors, one for each mode
approach	One of the approaches (for each relation in multi-relational networks in a vector) described in Žibera (2006). Possible values are: "bin" - binary blockmodeling,

"val" - valued blockmodeling,  
 "imp" - implicit blockmodeling,  
 "ss" - sum of squares homogeneity blockmodeling, and  
 "ad" - absolute deviations homogeneity blockmodeling.

...

Several other arguments, which are explained below. They are actually used by the function `gen.crit.fun`, however since this function is not intended to be called directly, it also has no help files. Therefore these arguments are described below. Which are needed depends on the approach selected:

**blocks:** A vector with names of allowed blocktypes. For multi-relational networks, it can be a list of such vectors. For approaches "bin", and "val", at least two should be selected. Possible values are are:

"null" - null or empty block  
 "com" - complete block  
 "rdo", "cdo" - row and column-dominant blocks (binary, valued, and implicit approach only)  
 "reg" - (f-)regular block  
 "rre", "cre" - row and column-(f-)regular blocks  
 "rfn", "cfn" - row and column-dominant blocks (binary, valued, and implicit approach only)  
 "den" - density block (binary approach only)  
 "avg" - average block (valued and implicit approach only)  
 "dnc" - do not care block - the error is always zero

The ordering is important, since if several block types have identical error, the first on the list is selected.

**BLOCKS:** An alternative to `blocks`. A pre-specified blockmodel. An array with dimensions three dimensions (see example below). The second and the third represent the clusters (for rows and columns), while the first is as long as the maximum number of allowed block types for a given block. If some block has less possible block types, the empty slots should have values `NA`. The values in the array should be the ones from above. For multi-relational networks, it can be a list of such arrays.

**m:** Sufficient value for individual cells for valued approach. Can be a number or a character string giving the name of a function. Set to "max" for implicit approach. For multi-relational networks, it can be a vector of such values.

**cut:** (default =  $\min(M[M > 0])$ ) The threshold used for binerizing the network for use with binary blockmodeling. All values with values lower than `cut` are recoded into 0s, all other into 1s. For multi-relational networks, it can be a vector of such values.

**FUN:** (default = "max") Function `f` used in row-f-regular, column-f-regular, and f-regular blocks. Not used in binary approach. For multi-relational networks, it can be a vector of such character strings.

**norm:** Should the block errors (inconsistencies) be normalized with the size

of the blocks, the block error does not depend on block size? The default is FALSE. Original version of implicit approach suggests TRUE, however the default is FALSE even for this approach based on better results in simulations. For multi-relational networks, it can be a vector.

normbym: The default is FALSE for valued and implicit approach, elsewhere not used. Original version of implicit approach suggests TRUE, however the default is FALSE even for this approach based on better results in simulations. For multi-relational networks, it can be a vector.

allow.max0: Should the maximum that is the basis for calculation of inconsistencies in implicit blockmodeling be allowed to be 0. If FALSE, the maximum is in such case set to the maximum of the network (if maximum of a block is 0) or to the maximum of the block (if row or column maximum is 0) Used only in implicit blockmodeling. If TRUE, the inconsistency of an ideal null block is 0 for all block types. The default is FALSE if null blocks are included in the allowed blocks in at least one block and TRUE otherwise.

allow.dom0: Should the dominant row or column (in row- or column-dominant blocks) be allowed to be 0. Used only in implicit blockmodeling. The default is FALSE.

normMto2rel: Create two-relation network from one relational network through row and column normalization. The default is FALSE:

sameModel: Should we demand the same blockmodel for all relations. If set to TRUE, it demands that across all relations the ideal block on the same position in the matrix BLOCKS should be chosen. Usually, these positions are occupied by the same blocks. If not, use with caution. The default is the value of normMto2rel.

max.con.val: Should the largest values be censored, limited to (larger values set to) - reasonable values are: "non" - (the default) no transformation is done  
 "m" - (the default for implicit blockmodeling) the maximum value equals the value of the parameter m  
 numerical values (usually) larger than parameter m and lower than the maximum value in M.

mindim: (default = 2) Minimal dimension (number of rows or columns) demanded for row and column-dominant and -functional blocks.

mindimreg: (default = FALSE) Should the mindim argument also be used for (row or column)-(f)-regular blocks

blockWeights: Weights for each type of block used, if they are to be different across block types (see blocks above). It must be supplied in form of a named vector

`blockWeights = c(name.of.block.type1=weight,...)`

If some of the block types used are not listed, they are given weight 1.

`positionWeights`: weights for positions in the blockmodel (the dimensions must be the same as the error matrix). For now this is a matrix (two-dimensional) even for multi-relational networks.

`save.err.v`: (default = FALSE) Should the error vector for all allowed block types in each block be saved?

`BLOCK.CV`: An array with the same dimensions as `BLOCKS` of central values for pre-specified homogeneity (sum of squares and absolute deviations) approach. For multi-relational networks, it can be a list of such arrays.

`CV.use`: An array with the same dimensions as `BLOCKS.CV` with instructions how to treat these central values. For multi-relational networks, it can be a list of such arrays. Possible alternatives are:

"fixed" - the central value is fixed to the value specified in `BLOCKS.CV`.

"min" - the central value specified in `BLOCKS.CV` is the minimal possible central value for a block. The central value for the block is computed as the maximum of the value specified in `BLOCKS.CV` and the empirical value computed based on tie values in the block.

"max" - the central value specified in `BLOCKS.CV` is the maximal possible central value for a block. The central value for the block is computed as the minimum of the value specified in `BLOCKS.CV` and the empirical value computed based on tie values in the block.

"free" - the central value is free, the value specified in `BLOCKS.CV` is ignored. The central value for the block is computed as the empirical value computed based on tie values in the block.

`use.for`: (default = TRUE) Should FORTRAN subroutines be used where available (available for only very special cases, currently only for using "ss" approach and only complete blocks. If you are using such setting and some special features (these are not implemented in FORTRAN subroutines), it's safer to set it to FALSE, as the function may miss that these features are not implemented in FORTRAN subroutines and use them nevertheless, leading to wrong results.

`diag`: (default = TRUE) Should the special status of diagonal be acknowledged.

## Value

A list:

M	The matrix of the network analyzed
err	The error or inconsistency empirical network with the ideal network for a given blockmodel (model,approach,...) and partition
clu	The analyzed partition
E	Block errors by blocks
IM	The obtained image

BM	Block means by block - only for Homogeneity blockmodeling
ERR.V	If selected. The error vector of errors for all allowed block types by blocks. The dimensions are [rows, columns (relations - if more than 1)]. Each cell contains a list of errors by block types

### Author(s)

Aleš Žiberna

### References

ŽIBERNA, Aleš (2006): Generalized Blockmodeling of Valued Networks. *Social Networks*, Jan. 2007, vol. 29, no. 1, 105-126. <http://dx.doi.org/10.1016/j.socnet.2006.04.002>.

ŽIBERNA, Aleš. Direct and indirect approaches to blockmodeling of valued networks in terms of regular equivalence. *J. math. sociol.*, 2008, vol. 32, no. 1, 57-84. <http://www.informaworld.com/smpp/content?content=10.1080/00222500701790207>.

DOREIAN, Patrick, BATAGELJ, Vladimir, FERLIGOJ, Anuška (2005): Generalized blockmodeling. (*Structural analysis in the social sciences*, 25). Cambridge [etc.]: Cambridge University Press, 2005. XV, 384 p., ISBN 0-521-84085-6.

### See Also

[opt.par](#), [opt.random.par](#), [opt.these.par](#), [check.these.par](#), [plot.crit.fun](#)

### Examples

```
#generating a simple network corresponding to the simple Sum of squares
#structural equivalence with blockmodel:
# null com
# null null
n<-20
net<-matrix(NA,ncol=n,nrow=n)
clu<-rep(1:2,times=c(5,15))
tclu<-table(clu)
net[clu==1,clu==1]<-rnorm(n=tclu[1]*tclu[1],mean=0,sd=1)
net[clu==1,clu==2]<-rnorm(n=tclu[1]*tclu[2],mean=4,sd=1)
net[clu==2,clu==1]<-rnorm(n=tclu[2]*tclu[1],mean=0,sd=1)
net[clu==2,clu==2]<-rnorm(n=tclu[2]*tclu[2],mean=0,sd=1)

#computation of criterion function with the correct partition
res<-crit.fun(M=net,clu=clu,approach="ss",blocks="com")
res$err #the error is relatively small
res$BM #The block means are around 0 or 4
plot(res)

#computation of criterion function with random partition
clu.rnd<-sample(1:2,size=n,replace=TRUE)
res.rnd<-crit.fun(M=net,clu=clu.rnd,approach="ss",blocks="com")
res.rnd$err #the error is larger
res.rnd$BM #random block means
```

```

plot(res.rnd)

#adapt network for Valued blockmodeling with the same model
net[net>4]<-4
net[net<0]<-0

#computation of criterion function with the correct partition
res<-crit.fun(M=net,clu=clu,approach="val",
  blocks=c("null","com"),m=4)
res$err #the error is relatively small
res$IM
#The image corresponds to the one used for generation of
#the network
plot(res)

#computation of criterion function with random partition
res.rnd<-crit.fun(M=net,clu=clu.rnd,approach="val",
  blocks=c("null","com")
  , m=4)
res.rnd$err #the error is larger
res.rnd$IM #all blocks are probably null
plot(res.rnd)

```

---

find.m

*Computing the threshold*


---

## Description

The functions compute the maximum value of  $m/cut$  where a certain block is still classified as `alt.blocks` and not "null". The difference between `find.m` and `find.m2` is that `find.m` uses an optimizational approach and is faster and more precise than `find.m2`. However, `find.m` only supports regular ("reg") and complete ("com") as `alt.blocks`, while `find.m2` supports all block types. Also, `find.m` does not always work, especially if `cormat` is not "none".

## Usage

```

find.m(M, clu, alt.blocks = "reg", diag = !is.list(clu),
  cormat = "none", half = TRUE, FUN = "max")
find.m2(M, clu, alt.blocks = "reg", neval = 100, half = TRUE,
  ms = NULL, ...)
find.cut(M, clu, alt.blocks = "reg", cuts = "all", ...)

```

## Arguments

**M** A matrix representing the (usually valued) network. For now, only one-relational networks are supported. The network can have one or more modes (different kinds of units with no ties among themselves). If the network is not two-mode, the matrix must be square.

clu	A partition. Each unique value represents one cluster. If the network is one-mode, than this should be a vector, else a list of vectors, one for each mode
alt.blocks	Only one of allowed blocktypes, as alternative to the null block: "com" - complete block "rdo", "cdo" - row and column-dominant blocks (binary, valued, and implicit approach only) "reg" - (f-)regular block "rre", "cre" - row and column-(f-)regular blocks "rfn", "cfn" - row and column-dominant blocks (binary, valued, and implicit approach only) "den" - density block (binary approach only) "avg" - average block (valued approach only)
diag	(default = TRUE) Should the special stauts of diagonal be acknowledged.
cormet	Which metho should be used to correct for diferent maxsimum error contributins? "none" - no correction "sensor" - censor values larger than m "correct" - so that the maxsimum possible error contribution of the cell is the same regardles of a condition (either that something must be o or at least m)
FUN	(default = "max") Function f used in row-f-regular, column-f-regular, and f-regular blocks.
cuts	The cuts which should be evaluatated. If cuts="all"n (default), all unique values are evaluated
neval	Number of different m values to be evaluated.
half	Should the returned value of m be one half of the value where the incosnistencies are the same.
ms	The values of m where the function should be evaluated.
...	Other parameters to crit.fun

### Value

A matrix of maximal m/cut values.

### Author(s)

Aleš Žiberna

### References

ŽIBERNA, Aleš (2006): Generalized Blockmodeling of Valued Networks. *Social Networks*, Jan. 2007, vol. 29, no. 1, 105-126. <http://dx.doi.org/10.1016/j.socnet.2006.04.002>.

ŽIBERNA, Aleš. Direct and indirect approaches to blockmodeling of valued networks in terms of regular equivalence. *J. math. sociol.*, 2008, vol. 32, no. 1, 57-84. <http://www.informaworld.com/smpp/content?content=10.1080/00222500701790207>.

DOREIAN, Patrick, BATAGELJ, Vladimir, FERLIGOJ, Anuška (2005): Generalized blockmodeling. (*Structural analysis in the social sciences*, 25). Cambridge [etc.]: Cambridge University Press, 2005. XV, 384 p., ISBN 0-521-84085-6.

**See Also**

[crit.fun](#) and maybe also [opt.par](#), [plot.mat](#)

---

formatA

*A formatting function for numbers*

---

**Description**

Formats a vector or matrix of numbers so that all have equal length (digits). This is especially suitable for printing tables.

**Usage**

```
formatA(x, digits = 2, FUN = round, ...)
```

**Arguments**

<code>x</code>	A numerical vector or matrix
<code>digits</code>	The number of desired digits
<code>FUN</code>	Function used for "shortening" the numbers.
<code>...</code>	Additional arguments to format

**Value**

A character vector or matrix.

**Author(s)**

Aleš Žiberna

**See Also**

[find.m](#), [find.m2](#), [find.cut](#)

**Examples**

```
A<-matrix(c(1,1.02002,0.2,10.3),ncol=2)
formatA(A)
```

fun.by.blocks

*Computation of function values by blocks***Description**

Computes a value of a functions over blocks of a matrix, defined by a partition.

**Usage**

```
fun.by.blocks(x, ...)

## Default S3 method:
fun.by.blocks(x = M, M = x, clu,
  ignore.diag = identical(ss(diag(M)), 0) && !is.list(clu),
  FUN = "mean", sortNames = TRUE, ...)

## S3 method for class 'opt.more.par'
fun.by.blocks(x, which = 1, ...)
```

**Arguments**

x	An object of suitable class or a matrix representing the (usually valued) network. For now, only one-relational networks are supported. The network can have one or more modes (diferent kinds of units with no ties among themselvs. If the network is not two-mode, the matrix must be square.
M	A matrix representing the (usually valued) network. For now, only one-relational networks are supported. The network can have one or more modes (diferent kinds of units with no ties among themselvs. If the network is not two-mode, the matrix must be square.
clu	A partition. Each unique value represents one cluster. If the network is one-mode, than this should be a vector, else a list of vectors, one for each mode
ignore.diag	Should the diagonal be ingored.
sortNames	Should the rows and columns of the matrix be sorted based on their names?
FUN	Function to be computed over the blocks
which	Which (if several) of the "best" solutions should be used
...	Further arguments to fun.by.blocks.default

**Value**

A numerical matrix of FUN values by blocks, induced by a partition clu

**Author(s)**

Aleš Žiberna

## References

ŽIBERNA, Aleš (2006): Generalized Blockmodeling of Valued Networks. *Social Networks*, Jan. 2007, vol. 29, no. 1, 105-126. <http://dx.doi.org/10.1016/j.socnet.2006.04.002>.

ŽIBERNA, Aleš. Direct and indirect approaches to blockmodeling of valued networks in terms of regular equivalence. *J. math. sociol.*, 2008, vol. 32, no. 1, 57-84. <http://www.informaworld.com/smpp/content?content=10.1080/00222500701790207>.

## See Also

[opt.random.par](#), [opt.these.par](#)

## Examples

```
n<-8 #if larger, the number of partitions increases dramatically,
    #as does if we increase the number of clusters
net<-matrix(NA,ncol=n,nrow=n)
clu<-rep(1:2,times=c(3,5))
tclu<-table(clu)
net[clu==1,clu==1]<-rnorm(n=tclu[1]*tclu[1],mean=0,sd=1)
net[clu==1,clu==2]<-rnorm(n=tclu[1]*tclu[2],mean=4,sd=1)
net[clu==2,clu==1]<-rnorm(n=tclu[2]*tclu[1],mean=0,sd=1)
net[clu==2,clu==2]<-rnorm(n=tclu[2]*tclu[2],mean=0,sd=1)

#we select a random partition and then optimise it

all.par<-nkpartitions(n=n, k=length(tclu)) #forming the partitions
all.par<-lapply(apply(all.par,1,list),function(x)x[[1]])
# to make a list out of the matrix

#optimizing 10 random partitions with opt.these.par
res<-opt.these.par(M=net,
  partitions=all.par[sample(1:length(all.par),size=10)],
  approach="ss", blocks="com")
plot(res) #Hopefully we get the original partition
fun.by.blocks(res)
#computing mean by blocks, ignoring the diagonal (default)
res$best[[1]]$BM
#the same result computed by opt.these.par when
#approach="ss" and blocks="com"
```

---

genRandomPar

*The function for generating random partitions*

---

## Description

The function generates random partitions. The function is meant to be called by the function [opt.random.par](#)

**Usage**

```
genRandomPar(k, n, seed = NULL, mingr = 1, maxgr = Inf,  
  addParam = list(genPajekPar = TRUE, probGenMech = NULL))
```

**Arguments**

k	Number of clusters (by modes)
n	Number of units (by modes)
seed	Seed for generating random numbers (partitions)
mingr	Minimal allowed group size
maxgr	Maximal allowed group size
addParam	This has to be a list with the following parameters (any or all can be missing, then the default values (see usage) are used): "genPajekPar" - Should the partitions be generated as in Pajek (Batagelj and Mrvar, 2006). If FALSE, all partitions are selected completely at random while making sure that the partitions have the required number of clusters. "probGenMech" - Here the probabilities for 4 different generating mechanisms can be specified. If this is not specified, the value is set to $c(1/3, 1/3, 1/3, 0)$ if genPajekPar is TRUE and to $c(0, 0, 0, 1)$ if genPajekPar is FALSE. The first 3 mechanisms are the same as implemented in Pajek (the second one has almost all units in only one cluster) and the fourth is completely random (from uniform distribution).

**Value**

A random partition in the format required by [opt.random.par](#). If a network has several modes, then a list of partitions, one for each mode.

**Author(s)**

Aleš Žiberna

**References**

BATAGELJ, Vladimir, MRVAR, Andrej (2006): Pajek 1.11, <http://vlado.fmf.unilj.si/pub/networks/pajek/> (accessed January 6, 2006).

**See Also**

[opt.random.par](#)

---

`gplot1`                      *A wrapper for function `gplot` - Two-Dimensional Visualization of Graphs*

---

### Description

The function calls function `gplot` from library `sna` with different defaults. Usefun for plotting image graphs.

### Usage

```
gplot1(M, diag = TRUE, displaylabels = TRUE, boxed.labels = FALSE,
       loop.cex = 4, arrowhead.cex = NULL, arrowheads.fun = "sqrt",
       edge.lwd = 1, edge.col = "default", rel.thresh = 0.05, ...)
gplot2(M, uselen = TRUE, usecurve = TRUE, edge.len = 0.001,
       diag = TRUE, displaylabels = TRUE, boxed.labels = FALSE,
       loop.cex = 4, arrowhead.cex = 2.5, edge.lwd = 1,
       edge.col = "default", rel.thresh = 0.05, ...)
```

### Arguments

<code>M</code>	A matrix (array) of a graph or set thereof. This data may be valued.
<code>diag</code>	boolean indicating whether or not the diagonal should be treated as valid data. Set this true if and only if the data can contain loops. <code>diag</code> is FALSE by default.
<code>rel.thresh</code>	real number indicating the lower relative (compared to the highest value) threshold for tie values. Only ties of value $>$ thresh are displayed. By default, thresh=0.
<code>displaylabels</code>	boolean; should vertex labels be displayed?
<code>boxed.labels</code>	boolean; place vertex labels within boxes?
<code>arrowhead.cex</code>	An expansion factor for edge arrowheads.
<code>arrowheads.fun</code>	A function for scaling arrowheads.
<code>loop.cex</code>	expansion factor for loops; may be given as a vector, if loops are to be of different sizes.
<code>edge.col</code>	color for edges; may be given as a vector or adjacency matrix, if edges are to be of different colors.
<code>edge.lwd</code>	line width scale for edges; if set greater than 0, edge widths are scaled by <code>edge.lwd*dat</code> . May be given as a vector or adjacency matrix, if edges are to have different line widths.
<code>edge.len</code>	if <code>uselen==TRUE</code> , curved edge lengths are scaled by <code>edge.len</code> .
<code>uselen</code>	boolean; should we use <code>edge.len</code> to rescale edge lengths?
<code>usecurve</code>	boolean; should we use <code>edge.curve</code> ?
<code>...</code>	additional arguments to <code>plot</code> or <code>gplot</code> from package <code>sna</code> :
	<code>mode</code> : the vertex placement algorithm; this must correspond to a <code>gplot.layout</code> function from package <code>sna</code> .

**Value**

Plots a graph

**Author(s)**

Aleš Žiberna

**See Also**

sna:gplot

---

ircNorm	<i>Function for iterated row and column normalization of valued matrices.</i>
---------	---

---

**Description**

The aim is to obtain a matrix with row and column sums equal to 1. This is achieved by iterating row and column normalization. This is usually not possible if any row or column has only 1 non-zero cell.

**Usage**

```
ircNorm(M, eps = 10^-12, maxiter = 1000)
```

**Arguments**

M	A non-negative valued matrix to be normalized
eps	The maximum allowed squared deviation of a row or column maximum from 1 (if not exactly 0). Also, if the all deviations in consecutive iterations are smaller, the process is terminated.
maxiter	Maximum number of iterations. If reached the process is terminated and the current solution returned

**Value**

Normalized matrix.

**Author(s)**

Aleš Žiberna

**Examples**

```
A<-matrix(runif(100),ncol=10)
A #A non-normalized matrix with different row and column sums.
apply(A,1,sum)
apply(A,2,sum)
A.norm<-ircNorm(A)
A.norm #Normalized matrix with all row and column sums aproximately 1.
apply(A.norm,1,sum)
apply(A.norm,2,sum)
```

---

nkpartitions	<i>Functions for listing all possible partitions or just counting the number of them.</i>
--------------	---

---

**Description**

The function nkpartitions lists all possible partitions of n objects in to k clusters. The function nkpar only gives the number of such partitions.

**Usage**

```
nkpartitions(n, k, exact = TRUE, print = FALSE)
nkpar(n, k)
```

**Arguments**

n	Number of units/objects
k	Number of clusters/groups
exact	Search for partitions with exactly k or at most k clusters
print	print results as they are found?

**Value**

The matrix or number of possible partitions.

**Author(s)**

Chris Andrews

**Examples**

```
n<-8 #if larger, the number of partitions increases dramatically,
      #as does if we increase the number of clusters
net<-matrix(NA,ncol=n,nrow=n)
clu<-rep(1:2,times=c(3,5))
tclu<-table(clu)
net[clu==1,clu==1]<-rnorm(n=tclu[1]*tclu[1],mean=0,sd=1)
```

```

net[clu==1,clu==2]<-rnorm(n=tclu[1]*tclu[2],mean=4,sd=1)
net[clu==2,clu==1]<-rnorm(n=tclu[2]*tclu[1],mean=0,sd=1)
net[clu==2,clu==2]<-rnorm(n=tclu[2]*tclu[2],mean=0,sd=1)

#computation of criterion function with the correct partition
nkpar(n=n, k=length(tclu)) #computing the number of partitions
all.par<-nkpartitions(n=n, k=length(tclu)) #forming the partitions
all.par<-lapply(apply(all.par,1,list),function(x)x[[1]])
# to make a list out of the matrix
res<-check.these.par(M=net,partitions=all.par,approach="ss",blocks="com")
plot(res) #we get the original partition

```

opt.par

*Optimizes a partition based on the value of a criterion function.***Description**

The function optimizes a partition based on the value of a criterion function (see [crit.fun](#)) for a given network and blockmodel for Generalized blockmodeling (Žiberna, 2006) based on other parameters (see below). The optimization is done through local optimization, where the neighbourhood of a partition includes all partitions that can be obtained by moving one unit from one cluster to another or by exchanging two units (from different clusters).

**Usage**

```

opt.par(M, clu, approach, ..., maxiter = 50, trace.iter =
  FALSE, switch.names = NULL, save.initial.param = TRUE,
  skip.par = NULL, save.checked.par =
  !is.null(skip.par), merge.save.skip.par =
  all(!is.null(skip.par), save.checked.par), check.skip
  = "never")

```

**Arguments**

M	A matrix representing the (usually valued) network. For now, only one-relational networks are supported. The network can have one or more modes (different kinds of units with no ties among themselves. If the network is not two-mode, the matrix must be square.
clu	A partition. Each unique value represents one cluster. If the network is one-mode, than this should be a vector, else a list of vectors, one for each mode
maxiter	Maximum number of iterations allowed
approach	One of the approaches described in Žiberna (2006). Possible values are: "bin" - binary blockmodeling, "val" - valued blockmodeling, "imp" - implicit blockmodeling, "ss" - sum of squares homogeneity blockmodeling, and "ad" - absolute deviations homogeneity blockmodeling.

... Argumets passed to other functions, see `crit.fun` and arguments to function `gen.crit.fun` (as this function is not intended to be called directly, it also has no help files). Some might be obligatory, e.g. argument `m` when using Valued blockmodeling approach. Therefore these arguments are described below:

`use.for.opt`: Should FORTRAN function be used for optimization if possible. If FORTRAN function is used, the speed is dramatically increast, however some the output is slightly different and the plotting function might not work. FORTRAN subrutines are available for only very special cases, currently only for using "ss" aproach and only complete blocks. If you are using such setting and some special features (these are not implemented in FORTRAN subrutines - e.g. using function `parOK` to allow only certain kinds of partitions), it's safer to set it to FASLE, as the fuction may miss that these features are not implemented in FORTRAN subrutines and use them nevertheless, leading to wrong results.

`use.for`: (default = TRUE) Should FORTRAN subrutines be used where available (available for only very special cases, currently only for using "ss" aproach and only complete blocks. If you are using such setting and some special features (these are not implemented in FORTRAN subrutines), it's safer to set it to FASLE, as the fuction may miss that these features are not implemented in FORTRAN subrutines and use them nevertheless, leading to wrong results.

`check.switch`: If TRUE (the default), the neighborhood of the selected partition also includes the partitions that can be obtained by exchanging (switching) two units from diferent clusters).

`check.all`: If TRUE (the default), all partitions in the neighborhood of the selected partition are first evaluated and the current partition than changes to the one with the lowest value of the criterion function (if lower than that of the current partition). If FALSE, the first partition with the criterion lower the current partition becomes the new current partition (and the iteration terminates).

<code>trace.iter</code>	Should the result of each iteration (and not only of the best one) be saved
<code>switch.names</code>	Should partitions that differ only in diferent names of positions be treated as different. It should be set to TRUE only if a asymmetric blockmodel via BLOCKS is specified. The default NULL tries to find that.
<code>save.initial.param</code>	Should the inital parameters (approach,...) be saved
<code>skip.par</code>	The partitions that are not allowed or were already checked and should therfire be skiped.
<code>save.checked.par</code>	Should the checked partitions be saved. For example, so that they can be used in the next call as <code>skip.par</code>
<code>merge.save.skip.par</code>	Should the checked partitions be merged with skiped ones?
<code>check.skip</code>	When should the check be preformed: "all" - before every call to 'crit.fun'

"iter" - at the end of each iteration  
 "never" - never

### Value

M	The matrix of the network analyzed
best	A list of results from <code>crit.fun.tmp</code> with the same elements as the result of <code>crit.fun</code> , only without M
iter	A list of results the same as <code>best</code> - one best for each iteration
err	If selected - The vector of errors or inconsistencies of the empirical network with the ideal network for a given blockmodel ( <code>model,approach,...</code> ) and partitions
nIter	The number of iterations used. It can show that <code>maxiter</code> is too low if this value is equal to <code>maxiter</code>
call	The call used to call the function.
initial.param	If selected - The initial parameters used.
checked.par	If selected - A list of checked partitions. If <code>merge.save.skip.par</code> is TRUE, this list also includes the partitions in <code>skip.par</code> .
...	

### Warning

This function can be extremely slow. The time complexity is increasing with the number of units and the number of clusters. It is advisable to first time the function on a smaller network.

### Author(s)

Aleš Žiberna

### References

ŽIBERNA, Aleš (2006): Generalized Blockmodeling of Valued Networks. *Social Networks*, Jan. 2007, vol. 29, no. 1, 105-126. <http://dx.doi.org/10.1016/j.socnet.2006.04.002>.

ŽIBERNA, Aleš. Direct and indirect approaches to blockmodeling of valued networks in terms of regular equivalence. *J. math. sociol.*, 2008, vol. 32, no. 1, 57-84. <http://www.informaworld.com/smp/content?content=10.1080/00222500701790207>.

DOREIAN, Patrick, BATAGELJ, Vladimir, FERLIGOJ, Anuška (2005): Generalized blockmodeling. (*Structural analysis in the social sciences*, 25). Cambridge [etc.]: Cambridge University Press, 2005. XV, 384 p., ISBN 0-521-84085-6.

### See Also

[crit.fun](#), [check.these.par](#), [opt.random.par](#), [opt.these.par](#), [plot.opt.par](#)

## Examples

```
n<-8 #if larger, the number of partitions increases dramatically,
      #as does if we increase the number of clusters
net<-matrix(NA,ncol=n,nrow=n)
clu<-rep(1:2,times=c(3,5))
tclu<-table(clu)
net[clu==1,clu==1]<-rnorm(n=tclu[1]*tclu[1],mean=0,sd=1)
net[clu==1,clu==2]<-rnorm(n=tclu[1]*tclu[2],mean=4,sd=1)
net[clu==2,clu==1]<-rnorm(n=tclu[2]*tclu[1],mean=0,sd=1)
net[clu==2,clu==2]<-rnorm(n=tclu[2]*tclu[2],mean=0,sd=1)

#we select a random partition and then optimise it

all.par<-nkpartitions(n=n, k=length(tclu)) #forming the partitions
all.par<-lapply(apply(all.par,1,list),function(x)x[[1]])
# to make a list out of the matrix
res<-opt.par(M=net,
             clu=all.par[[sample(1:length(all.par),size=1)]],
             approach="ss",blocks="com")
plot(res) #Hopefully we get the original partition
```

---

opt.random.par

*Optimizes a set of partitions based on the value of a criterion function.*

---

## Description

The function optimizes a set partitions based on the value of a criterion function (see [crit.fun](#) for details on the criterion function) for a given network and blockmodel for Generalized blockmodeling (Žibera, 2006) based on other parameters (see below). The optimization is done through local optimization, where the neighborhood of a partition includes all partitions that can be obtained by moving one unit from one cluster to another or by exchanging two units (from different clusters). A list of partitions can be specified (opt.these.par) or the number of clusters and a number of partitions to generate (opt.random.par).

## Usage

```
opt.random.par(M, k, n = NULL, rep, approach, ..., return.all =
FALSE, return.err = TRUE, maxiter = 50, trace.iter =
FALSE, switch.names = NULL, save.initial.param = TRUE,
skip.par = NULL, save.checked.par = TRUE,
merge.save.skip.par = any(!is.null(skip.par),
save.checked.par), skip.allready.checked.par = TRUE,
check.skip = "iter", print.iter = FALSE, max.iden =
10, seed = NULL, parGenFun = genRandomPar, mingr = 1,
maxgr = Inf, addParam = list(genPajekPar = TRUE,
probGenMech = NULL), maxTriesToFindNewPar = rep * 10)

opt.these.par(M, partitions, approach, ..., return.all = FALSE,
```

```

return.err = TRUE, skip.allready.checked.par = TRUE,
maxiter = 50, trace.iter = FALSE, switch.names = NULL,
save.initial.param = TRUE, skip.par = NULL,
save.checked.par = !is.null(skip.par),
merge.save.skip.par = all(!is.null(skip.par),
save.checked.par), check.skip = "never", print.iter =
FALSE)

```

## Arguments

M	A matrix representing the (usually valued) network. For now, only one-relational networks are supported. The network can have one or more modes (different kinds of units with no ties among themselves. If the network is not two-mode, the matrix must be square.
k	The number of clusters used in generation of partitions.
n	The vector of the number of units in each mode (only necessary if mode is larger than 2).
rep	The number of repetitions/different starting partitions to check.
partitions	A list of partitions. Each unique value represents one cluster. If the network is one-mode, then this should be a vector, else a list of vectors, one for each mode.
approach	One of the approaches described in Žiberna (2007). Possible values are: "bin" - binary blockmodeling, "val" - valued blockmodeling, "imp" - implicit blockmodeling, "ss" - sum of squares homogeneity blockmodeling, and "ad" - absolute deviations homogeneity blockmodeling.
...	Arguments passed to other functions, see <a href="#">crit.fun</a> and arguments to function <code>gen.crit.fun</code> (as this function is not intended to be called directly, it also has no help files). Some might be obligatory, e.g. argument <code>m</code> when using Valued blockmodeling approach. Therefore these arguments are described below:

`use.for.opt`: Should FORTRAN function be used for optimization if possible. If FORTRAN function is used, the speed is dramatically increased, however some the output is slightly different and the plotting function might not work. FORTRAN subroutines are available for only very special cases, currently only for using "ss" approach and only complete blocks. If you are using such setting and some special features (these are not implemented in FORTRAN subroutines - e.g. using function `parOK` to allow only certain kinds of partitions), it's safer to set it to `FASLE`, as the function may miss that these features are not implemented in FORTRAN subroutines and use them nevertheless, leading to wrong results.

`use.for`: (default = TRUE) Should FORTRAN subroutines be used where available (available for only very special cases, currently only for using "ss" approach and only complete blocks. If you are using such setting and some special features (these are not implemented in FORTRAN subroutines), it's safer to set it to `FASLE`, as the function may miss that these features are not implemented in FORTRAN subroutines and use them nevertheless, leading to wrong results.

check.switch: If TRUE (the default), the neighborhood of the selected partition also includes the partitions that can be obtained by exchanging (switching) two units from different clusters).

check.all: If TRUE (the default), all partitions in the neighborhood of the selected partition are first evaluated and the current partition then changes to the one with the lowest value of the criterion function (if lower than that of the current partition). If FALSE, the first partition with the criterion lower than the current partition becomes the new current partition (and the iteration terminates).

return.all	If FALSE, solution for only the best (one or more) partition/s is/are returned.
return.err	Should the error for each optimized partition be returned
maxiter	Maximum number of iterations
trace.iter	Should the result of each iteration (and not only of the best one) be saved
switch.names	Should partitions that differ only in different names of positions be treated as different. It should be set to TRUE only if an asymmetric blockmodel via BLOCKS is specified.
save.initial.param	Should the initial parameters (approach,...) be saved
skip.par	The partitions that are not allowed or were already checked and should therefore be skipped.
save.checked.par	Should the checked partitions be saved. For example, so that they can be used in the next call as skip.par
merge.save.skip.par	Should the checked partitions be merged with skipped ones?
skip.already.checked.par	If TRUE, the partitions that were already checked when running opt.par from different starting points will be skipped.
check.skip	When should the check be performed: "all" - before every call to 'crit.fun' (Time demanding) "iter" - at the end of each iteration "opt.par" - before every call to opt.par, when starting the optimization of a new partition. "never" - never
print.iter	Should the progress of each iteration be printed?
max.iden	The maximum number of results that should be saved (in case there are more than max.iden results with minimal error, only the first max.iden will be saved).
seed	Optional. The seed for random generation of partitions.
parGenFun	The function (object) that will generate random partitions. The default function is <a href="#">genRandomPar</a> . The function has to accept the following parameters: k (number of partitions by modes), n (number of units by modes), seed (seed value for random generation of partition), addParam (a list of additional parameters)

mingr	Minimal allowed group size
maxgr	Maximal allowed group size
addParam	A list of additional parameters for function specified above. In the usage section they are specified for the default function <a href="#">genRandomPar</a> :
maxTriesToFindNewPar	The maximum number of partition try when trying to find a new partition to optimize that was not yet checked before - the default value is rep*1000

### Value

M	The matrix of the network analyzed
res	If <code>return.all = TRUE</code> - A list of results the same as <code>best</code> - one best for each partition optimized
best	A list of results from <code>crit.fun.tmp</code> with the same elements as the result of <code>crit.fun</code> , only without M
err	If <code>return.err = TRUE</code> - The vector of errors or inconsistencies of the empirical network with the ideal network for a given blockmodel (model, approach,...) and partitions
nIter	The vector of number of iterations used - one value for each starting partition that was optimized. It can show that <code>maxiter</code> is too low if a lot of these values have the value of <code>maxiter</code>
checked.par	If selected - A list of checked partitions. If <code>merge.save.skip.par</code> is <code>TRUE</code> , this list also includes the partitions in <code>skip.par</code> .
call	The call used to call the function.
initial.param	If selected - The initial parameters used.

### Warning

This function can be extremely slow. The time complexity is increasing with the number of units and the number of clusters. It is advisable to first time the function on a smaller network.

### Author(s)

Aleš Žiberna

### References

- ŽIBERNA, Aleš (2007): Generalized Blockmodeling of Valued Networks. *Social Networks*, Jan. 2007, vol. 29, no. 1, 105-126. <http://dx.doi.org/10.1016/j.socnet.2006.04.002>.
- ŽIBERNA, Aleš. Direct and indirect approaches to blockmodeling of valued networks in terms of regular equivalence. *J. math. sociol.*, 2008, vol. 32, no. 1, 57-84. <http://www.informaworld.com/smp/content?content=10.1080/00222500701790207>.
- DOREIAN, Patrick, BATAGELJ, Vladimir, FERLIGOJ, Anuška (2005): Generalized blockmodeling. (*Structural analysis in the social sciences*, 25). Cambridge [etc.]: Cambridge University Press, 2005. XV, 384 p., ISBN 0-521-84085-6.
- BATAGELJ, Vladimir, MRVAR, Andrej (2006): Pajek 1.11, <http://vlado.fmf.unilj.si/pub/networks/pajek/> (accessed January 6, 2006).

**See Also**

[crit.fun](#), [check.these.par](#), [opt.par](#), [plot.opt.more.par](#)

**Examples**

```
n<-8 #if larger, the number of partitions increases dramatically,
      #as does if we increase the number of clusters
net<-matrix(NA,ncol=n,nrow=n)
clu<-rep(1:2,times=c(3,5))
tclu<-table(clu)
net[clu==1,clu==1]<-rnorm(n=tclu[1]*tclu[1],mean=0,sd=1)
net[clu==1,clu==2]<-rnorm(n=tclu[1]*tclu[2],mean=4,sd=1)
net[clu==2,clu==1]<-rnorm(n=tclu[2]*tclu[1],mean=0,sd=1)
net[clu==2,clu==2]<-rnorm(n=tclu[2]*tclu[2],mean=0,sd=1)

#we select a random partition and then optimise it

all.par<-nkpartitions(n=n, k=length(tclu))
#forming the partitions
all.par<-lapply(apply(all.par,1,list),function(x)x[[1]])
# to make a list out of the matrix

#optimizing one partition
res<-opt.par(M=net,
             clu=all.par[[sample(1:length(all.par),size=1)]],
             approach="ss",blocks="com")
plot(res) #Hopefully we get the original partition

#optimizing 10 random chosen partitions with opt.these.par
res<-opt.these.par(M=net,
                  partitions=all.par[sample(1:length(all.par),size=10)],
                  approach="ss", blocks="com")
plot(res) #Hopefully we get the original partition

#optimizing 10 random chosen partitions with opt.random.par
res<-opt.random.par(M=net,k=2,rep=10,approach="ss",blocks="com")
plot(res) #Hopefully we get the original partition
```

**Description**

Functions for reading/loading and writing Pajek files:

loadnetwork - Loads a Pajek ".net" filename as a matrix. For now, only simple one and two-mode networks are supported (eg. only single relations, no time information).

loadnetwork2 - The same as above, but adopted to be called withih loadpajek

loadvector - Loads a Pajek ".clu", ".vec" or ".per" file as a vector.

loadvector2 - The same as above, but adopted to be called withih loadpajek - as a consequence not suited for reading clusters

loadmatrix - Loads a Pajek ".mat" file as a matrix.

loadpajek - Loads a Pajek project filename (".paj") as a list with the following components: Networks, Partitions, Vectors and Clusters. Clusters and hierarchies are dismissed.

savevector - Saves a vector, permutation or partition to a Pajek ".clu", ".vec" or ".per" file.

savenetwork - Saves a matrix in to a Pajek ".net" file

savematrix - Saves a matrix in to a Pajek ".mat" file

savecluster - Saves a vector to a Pajek ".cls" file.

savepajek - Saves a list of objects to a Pajek ".paj" file.

### Usage

```
loadnetwork(filename,useSparseMatrix=NULL,minN=50)
loadnetwork2(filename,useSparseMatrix=NULL,minN=50)
loadmatrix(filename)
loadvector(filename)
loadvector2(filename)
loadpajek(filename)
```

```
savenetwork(n, filename, twomode = "default", symmetric = NULL, cont=FALSE)
savematrix(n, filename, twomode = 1, cont=FALSE)
savevector(v, filename, cont=FALSE)
savecluster(v, filename, cont=FALSE)
savepajek(pajekList,filename,twomode="default",asMatrix=FALSE,symmetric=NULL)
```

### Arguments

filename	The name of the filename to be loaded or saved to or an open file object.
useSparseMatrix	Should a sparse matrix be use instead of the ordinary one? Sparse matices can only be used if package Matrix is installed. The default NULL uses sparsematrices for networks with more that minN vertices
minN	The minimal number of units in the network to use sparse matrices.
n	A matrix representing the network.
twomode	1 for one-mode networks and 2 for two-mode networks. Default sets the argument to 1 for square matrices and to 2 for others.
symetric	If true, only the lower part of the matrix is used and the values are interpreted as "Edges", not "Arcs". When used in savepajek function it applies to all networks.
v	A vector
cont	A logical constant indicating if the output should be appended to a file instead of creating a new file. Included so that the function could be used within savepajek function. Not intended to be used/set by user.

pajekList	A list containing one or more lists with the following names (exact): "Networks", "Partitions", "Vectors", "Permutations" and "Clusters". Each list should contain elements that correspond to Pajek objects ("Networks" matrices and the other lists vectors), meaning that they can be written to Pajek files using the other save* functions . The content is written to Pajek ".paj" file.
asMatrix	A logical constant indicating if the networks should be written in matrix format (as in ".mat" file) instead in the network (*Arcslist or *Edgelist) format (as in ".mat" file).

**Value**

NULL, a matrix or a vector (see Description)

**Author(s)**

Vladimir Batagelj & Andrej Mrvar (most functions), Aleš Žiberna (loadnetwork, loadpajek and modification of others)

**References**

Pajek ( V. Batagelj, A. Mrvar: Pajek - Program for Large Network Analysis. Home page <http://vlado.fmf.uni-lj.si/pub/networks/pajek/>.

W. de Nooy, A. Mrvar, V. Batagelj: Exploratory Social Network Analysis with Pajek, CUP, January 2005

**See Also**

[plot.mat](#), [crit.fun](#), [opt.par](#), [opt.random.par](#), [opt.these.par](#), [check.these.par](#)

---

plot.mat

*Functions for plotting a partitioned matrix*

---

**Description**

The main function `plot.mat` plots a (optionally partitioned) matrix. If the matrix is partitioned, the rows and columns of the matrix are rearranged according to the partitions. Other functions are only wrappers for `plot.mat` for convenience when plotting the results of the corresponding functions. The `plot.mat.nm` plots two matrices based on `M`, normalized by rows and columns, next to each other.

**Usage**

```
plot.mat(x=M, M=x, clu = NULL, ylab = "", xlab = "", main = NULL,
  print.val = !length(table(M)) <= 2, print.0 = FALSE,
  plot.legend = !print.val && !length(table(M)) <= 2,
  print.legend.val = "out", print.digits.legend = 2,
  print.digits.cells = 2, print.cells.mf = NULL,
```

```

outer.title = !plot.legend,
title.line = ifelse(outer.title, -1.5, 7),
mar = c(0.5, 7, 8.5, 0) + 0.1, cex.val = "default",
val.y.coor.cor = 0, val.x.coor.cor = 0, cex.legend = 1,
legend.title = "Legend", cex.axes = "default",
print.axes.val = NULL,
print.x.axis.val = !is.null(colnames(M)),
print.y.axis.val = !is.null(rownames(M)),
x.axis.val.pos = 1.1, y.axis.val.pos = -0.1,
cex.main = par()$cex.main, cex.lab = par()$cex.lab,
yaxis.line = -1.5, xaxis.line = -1, legend.left = 0.4,
legend.up = 0.03, legend.size = 1/min(dim(M)),
legend.text.hor.pos = 0.5, par.line.width = 3,
par.line.col = "blue", IM.dens = NULL, IM = NULL, wnet = 1,
wIM = NULL,
use.IM = length(dim(IM))==length(dim(M))|!is.null(wIM),
dens.leg = c(null = 100),
blackdens = 70, plotLines = TRUE, ...)

plot.mat.nm(x=M, M=x, ..., main.title = NULL,
  title.row = "Row normalized",
  title.col = "Column normalized",
  main.title.line = -2, par.set = list(mfrow = c(1, 2)))

## S3 method for class 'mat'
plot(x=M, M=x, clu = NULL, ylab = "", xlab = "",
  main = NULL, print.val = !length(table(M)) <= 2,
  print.0 = FALSE,
  plot.legend = !print.val && !length(table(M)) <= 2,
  print.legend.val = "out", print.digits.legend = 2,
  print.digits.cells = 2, print.cells.mf = NULL,
  outer.title = !plot.legend,
  title.line = ifelse(outer.title, -1.5, 7),
  mar = c(0.5, 7, 8.5, 0) + 0.1, cex.val = "default",
  val.y.coor.cor = 0, val.x.coor.cor = 0, cex.legend = 1,
  legend.title = "Legend", cex.axes = "default",
  print.axes.val = NULL,
  print.x.axis.val = !is.null(colnames(M)),
  print.y.axis.val = !is.null(rownames(M)),
  x.axis.val.pos = 1.1, y.axis.val.pos = -0.1,
  cex.main = par()$cex.main, cex.lab = par()$cex.lab,
  yaxis.line = -1.5, xaxis.line = -1, legend.left = 0.4,
  legend.up = 0.03, legend.size = 1/min(dim(M)),
  legend.text.hor.pos = 0.5, par.line.width = 3,
  par.line.col = "blue", IM.dens = NULL, IM = NULL, wnet = 1,
  wIM = NULL,
  use.IM = length(dim(IM)) == length(dim(M)) | !is.null(wIM),
  dens.leg = c(null = 100), blackdens = 70, plotLines = TRUE,

```

```

... )

## S3 method for class 'crit.fun'
plot(x, main = NULL, ...)

## S3 method for class 'opt.par'
plot(x, main = NULL, which = 1, ...)

## S3 method for class 'opt.par.mode'
plot(x, main = NULL, which = 1, ...)

## S3 method for class 'opt.more.par'
plot(x, main = NULL, which = 1, ...)

## S3 method for class 'opt.more.par.mode'
plot(x, main = NULL, which = 1, ...)

## S3 method for class 'check.these.par'
plot(x, main = NULL, which = 1, ...)

```

### Arguments

x	A result from a coresponding function or a matrix or similar object representing a network
M	A matrix or similar object representing a network - either x or M must be supplied - both are here to make the code compatible with generic and with older functions
clu	A partition
ylab	Label for y axis
xlab	Label for x axis
main	Main title
main.title	Main title in nm version
main.title.line	The line in which main title is printed in nm version
title.row	Title for the row-normalized matrix in nm version
title.col	Title for the column-normalized matrix in nm version
par.set	A list of possible plotting paramters (to par) to be used in nm version
print.val	Should the values be printed in the matrix
print.0	If print.val=TRUE Should the 0s be printed in the matrix
plot.legend	Should the legend for shades be plotted
print.legend.val	Should the values be printed in the legend
print.digits.legend	The number of digits that should appear in the legend

print.digits.cells	The number of digits that should appear in the cells (of the matrix and/or legend)
print.cells.mf	if not NULL, the above argument is ignored, the cell values are printed as the cell are multiplied by this factor and rounded
outer.title	Should the title be printed on the 'inner' or 'outer' plot, default is 'inner' if legend is plotted and 'outer' otherwise. May be soon omitted.
title.line	The line (from the top) where the title should be printed. The suitable values depend heavily on the display type.
mar	A numerical vector of the form 'c(bottom, left, top, right)' which gives the lines of margin to be specified on the four sides of the plot. The R default for ordinary plots is 'c(5, 4, 4, 2) + 0.1', while this functions default is c(0.5, 7, 8.5, 0) + 0.1.
cex.val	Size of the values printed. The "default" is 10/"number of units"
val.y.coor.cor	Correction for centering the values in the squares in y direction
val.x.coor.cor	Correction for centering the values in the squares in x direction
cex.legend	Size of the text in the legend
legend.title	The title of the legend
cex.axes	Size of the characters in axes, 'default' makes the cex so small that all categories can be printed
print.axes.val	Should the axes values be printed, 'default' prints each axis if 'rownames' or 'colnames' is not 'NULL'
print.x.axis.val	Should the x axis values be printed, 'default' prints each axis if 'rownames' or 'colnames' is not 'NULL'
print.y.axis.val	Should the y axis values be printed, 'default' prints each axis if 'rownames' or 'colnames' is not 'NULL'
x.axis.val.pos	x coordinate of the y axis values
y.axis.val.pos	y coordinate of the x axis values
cex.main	Size of the text in the main title
cex.lab	Size of the text in matrix
yaxis.line	The position of the y axis (the argument 'line')
xaxis.line	The position of the x axis (the argument 'line')
legend.left	How much left should the legend be from the matrix
legend.up	How much up should the legend be from the matrix
legend.size	Relative legend size
legend.text.hor.pos	Horizontal position of the legend text (bottom) - 0 = bottom, 0.5 = middle,...
par.line.width	The width of the line that separates the partitions
par.line.col	The color of the line that separates the partitions
IM.dens	The density of shading lines for each block

IM	The image (as obtained with <code>crit.fun</code> ) of the blockmodel. <code>dens.leg</code> is used to translate this image into <code>IM.dens</code> .
<code>dens.leg</code>	It is used to translate the IM into <code>IM.dens</code> .
<code>blackdens</code>	At which density should the values on dark colours of lines be printed in white.
<code>plotLines</code>	Should the lines in the matrix be printed - default TRUE, best set to FALSE for larger networks.
<code>which</code>	Which (if there are more than one) of optimal solutions to plot
<code>wnet</code>	Specifies which net (if more) should be plotted - used if <code>M</code> is an array.
<code>wIM</code>	Specifies which IM (if more) should be used for plotting (default = <code>wnet</code> ) - used if <code>IM</code> is an array.
<code>use.IM</code>	Specifies if IM should IM be used for plotting? be used for plotting?
...	Additional arguments to <code>plot.default</code> for <code>plot.mat</code> and also to <code>plot.mat</code> for other functions

### Value

The functions are used for their side affect - plotting.

### Author(s)

Aleš Žiberna

### References

ŽIBERNA, Aleš (2006): Generalized Blockmodeling of Valued Networks. *Social Networks*, Jan. 2007, vol. 29, no. 1, 105-126. <http://dx.doi.org/10.1016/j.socnet.2006.04.002>.

ŽIBERNA, Aleš. Direct and indirect approaches to blockmodeling of valued networks in terms of regular equivalence. *J. math. sociol.*, 2008, vol. 32, no. 1, 57-84. <http://www.informaworld.com/smpp/content?content=10.1080/00222500701790207>.

### See Also

[crit.fun](#), [opt.par](#), [opt.random.par](#), [opt.these.par](#), [check.these.par](#)

### Examples

```
#Generation of the network
n<-20
net<-matrix(NA,ncol=n,nrow=n)
clu<-rep(1:2,times=c(5,15))
tclu<-table(clu)
net[clu==1,clu==1]<-rnorm(n=tclu[1]*tclu[1],mean=0,sd=1)
net[clu==1,clu==2]<-rnorm(n=tclu[1]*tclu[2],mean=4,sd=1)
net[clu==2,clu==1]<-rnorm(n=tclu[2]*tclu[1],mean=0,sd=1)
net[clu==2,clu==2]<-rnorm(n=tclu[2]*tclu[2],mean=0,sd=1)

#Ploting the network
plot.mat(M=net, clu=clu)
```

```
class(net)<-"mat"  
plot(net, clu=clu)  
#See coresponding functions for examples for other plotting  
#functions  
#presented, that are esentially only the wrappers for "plot.max"
```

---

rand

*Comparing partitions*

---

### Description

Rand Index and Rand Index corrected/adjusted for chance for comparing partitions (Hubert and Arabie, 1985). The names of the clusters do not matter.

### Usage

```
rand(tab)  
rand2(clu1, clu2)  
crand(tab)  
crand2(clu1, clu2)
```

### Arguments

clu1, clu2	The two partitions to be compared, given in the form of vectors, where for each unit a cluster membership is given.
tab	A contingency table obtained as <code>table(clu1,clu2)</code>

### Value

The value of Rand Index (corrected/adjusted for chance)

### Author(s)

Aleš Žiberna

### References

Hubert L. in Arabie P. (1985): Comparing Partitions. *Journal of Classification*, 2, 193-218.

---

recode	<i>Recode</i>
--------	---------------

---

**Description**

Recodes values in a vector.

**Usage**

```
recode(x, oldcode = sort(unique(x)), newcode)
```

**Arguments**

x	A vector
oldcode	A vector of old codes
newcode	A vector of new codes

**Value**

A recoded vector

**Author(s)**

Aleš Žiberna

**Examples**

```
x<-rep(1:3,times=1:3)
newx<-recode(x,oldcode=1:3,newcode=c("a","b","c"))
```

---

REGE	<i>REGE - Algorithms for computing (dis)similarities in terms of regular equivalence.</i>
------	---

---

**Description**

REGE - Algorithms for computing (dis)similarities in terms of regular equivalence (White and Reitz, 1983):

REGE, REGE.for - Classical REGE or REGGE, as also implemented in Ucinet. Similarities in terms of regular equivalence are computed. The REGE.for is a wrapper for calling the FORTRAN subroutine written by White (1985a), modified to be called by R. The REGE does the same, however it is written in R. The functions with and without ".for" differ only in whether they are implemented in R or FORTRAN. Needless to say, the functions implemented in FORTRAN are much faster.

REGE.ow, REGE.ow.for - The above function, modified so that a best match is searched for for each arc separately (and not for both arcs, if they exist, together)

REGE.nm.for - REGE or REGGE, modified to to use row and column normalited matrices instead of the original matrix.

REGE.ownm.for - The above function, modified so that a best match is searched for for each arc splerately (and not for both arcs, if they exist, together)

REGD.for - REGD or REGDI, a dissimilarity version of the classical REGE or REGGE. Dissimilarities in terms of regular equivalnece are computed. The REGD.for is a wrapper for calling the FORTRAN subrutine written by White (1985b), modified to be called by R.

REGE.FC - Acctually an erlier version of REGE. The diference is in the denominator. See Žiberna (2006) for details.

REGE.FC.ow - The above function, modified so that a best match is searched for for each arc splerately (and not for both arcs, if they exist, together)

other - still in testing stage

### Usage

```

REGE(M, E = 1, iter = 3, until.change = TRUE, use.diag = TRUE)
REGE.for(M, iter = 3, E = 1)
REGE.nm.for(M, iter = 3, E = 1)
REGE.ow(M, E = 1, iter = 3, until.change = TRUE, use.diag = TRUE)
REGE.ow.for(M, iter = 3, E = 1)
REGE.ownm.for(M, iter = 3, E = 1)
REGD.for(M, iter = 3, E = 0)
REGD.ow.for(M, iter = 3, E = 0)
REGE.FC(M, E = 1, iter = 3, until.change = TRUE, use.diag = TRUE,
normE = FALSE)
REGE.FC.ow(M, E = 1, iter = 3, until.change = TRUE,
use.diag = TRUE, normE = FALSE)
REGD.ne.for(M, iter = 3, E=0)
REGD.ow.ne.for(M, iter = 3, E = 0)
REGE.ne.for(M, iter = 3, E=1)
REGE.nm.diag.for(M, iter = 3, E=1)
REGE.nm.ne.for(M, iter = 3, E=1)
REGE.ow.ne.for(M, iter = 3, E=1)
REGE.ownm.diag.for(M, iter = 3, E=1)
REGE.ownm.ne.for(M, iter = 3, E=1)

```

### Arguments

M	Matrix or a 3 dimensional array representing the network. The third dimension allows for several relations to be analyzed.
E	Initial (dis)similarity in terms of regular equivalnece.
iter	The desired number of itetations
until.change	Should the iterations be stop when no change occurs
use.diag	Should the diagonal be used. If FALSE, all diagonal elements are set to 0.
normE	Should the equivalence matrix be normalized after each iteration?

**Value**

E	A matrix of (dis)similarities in terms of regular equivalence
Eall	An array of (dis)similarity matrices in terms of regular equivalence, each third dimension represents one iteration. For ".for" functions, only the initial and the final (dis)similarities are returned.
M	Matrix or a 3 dimensional array representing the network used in the call.
iter	The desired number of iterations
use.diag	Should the diagonal be used - for functions implemented in R only.
...	

**Author(s)**

Aleš Žiberna based on Douglas R. White's original REGE and REGD

**References**

- ŽIBERNA, Aleš. Direct and indirect approaches to blockmodeling of valued networks in terms of regular equivalence. *J. math. sociol.*, 2008, vol. 32, no. 1, 57-84. <http://www.informaworld.com/smpp/content?content=10.1080/00222500701790207>.
- White, D. R., K. P. Reitz (1983): "Graph and semigroup homomorphisms on networks of relations". *Social Networks*, 5, p. 193-234.
- White, Douglas R.(1985a): DOUG WHITE'S REGULAR EQUIVALENCE PROGRAM. <http://eclectic.ss.uci.edu/~drwhite/REGGE/REGGE.FOR> (12.5.2005).
- White, Douglas R.(1985b): DOUG WHITE'S REGULAR DISTANCES PROGRAM. <http://eclectic.ss.uci.edu/~drwhite/REGGE/REGDI.FOR> (12.5.2005).
- White, Douglas R.(2005): REGGE (web page). <http://eclectic.ss.uci.edu/~drwhite/REGGE/> (12.5.2005).

**See Also**

[sedist](#), [crit.fun](#), [opt.par](#), [plot.mat](#)

**Examples**

```
n<-20
net<-matrix(NA,ncol=n,nrow=n)
clu<-rep(1:2,times=c(5,15))
tclu<-table(clu)
net[clu==1,clu==1]<-0
net[clu==1,clu==2]<-rnorm(n=tclu[1]*tclu[2],
  mean=4,sd=1)*sample(c(0,1),
  size= tclu[1]*tclu[2],replace=TRUE,prob=c(3/5,2/5))
net[clu==2,clu==1]<-0
net[clu==2,clu==2]<-0

D<-REGE.for(M=net)$E #any other REGE function can be used
plot.mat(net, clu=cutree(hclust(d=as.dist(1-D),method="ward"),
```

```

      k=2))
#REGE returns similarities, which have to be converted to
#disimilarities

res<-opt.random.par(M=net,k=2,rep=10,approach="ss",blocks="reg",
  FUN="max")
plot(res) #Hopefully we get the original partition

```

---

reorderImage	<i>Reorders an image matrix of the blockmodel (or an error matrix based on new and old partition.</i>
--------------	---

---

### Description

Reorders an image matrix of the blockmodel (or an error matrix based on new and old partition. The partitions should be the same, except that classes can have different labels. It is useful when we want to have a different order of classes in figures and then also in image matrices. Currently it is only suitable for one-mode blockmodels

### Usage

```
reorderImage(IM, oldClu, newClu)
```

### Arguments

IM	An image or error matrix.
oldClu	Old partition
newClu	New partition, the same as the old one except for class labels.

### Value

Reorder matrix (rows and columns are reordered)

### Author(s)

Ales Ziberna

### References

ŽIBERNA, Aleš (2006): Generalized Blockmodeling of Valued Networks. *Social Networks*, Jan. 2007, vol. 29, no. 1, 105-126. <http://dx.doi.org/10.1016/j.socnet.2006.04.002>.

ŽIBERNA, Aleš. Direct and indirect approaches to blockmodeling of valued networks in terms of regular equivalence. *J. math. sociol.*, 2008, vol. 32, no. 1, 57-84. <http://www.informaworld.com/smpp/content?content=10.1080/00222500701790207>.

### See Also

[crit.fun](#), [plot.mat](#), [clu](#), [IM](#), [err](#)

---

sedist	<i>Computes distances in terms of Structural equivalence (Lorrain and White, 1971)</i>
--------	--

---

### Description

The functions computed the distances in terms of Structural equivalence (Lorrain and White, 1971) between the units of a one-mode network. Several options for treating the diagonal values are supported.

### Usage

```
sedist(M, method = "default", fun = "default",
      fun.on.rows = "default", handle.interaction = "switch",
      use = "pairwise.complete.obs", ...)
```

### Arguments

M	A matrix representing the (usually valued) network. For now, only one-relational networks are supported. The network must be one-mode.
method	The method used to compute distances - any of the methods allowed by functions <code>dist</code> , <code>cor</code> or <code>cov</code> (all <code>package::stats</code> ) or just "cor" or "cov" (given as character).
fun	Which function should be used to compute distances (given as character), .
fun.on.rows	For non-standard function - does the function compute measure on rows (such as <code>cor</code> , <code>cov</code> ,...) of the data matrix (as opposed to computing measure on columns (such as <code>dist</code> ).
handle.interaction	How should the interaction between the vertices analysed be handled: "switch" (the default) - assumes that when comparing units <i>i</i> and <i>j</i> , $M[i,i]$ should be compared with $M[j,j]$ and $M[i,j]$ with $M[j,i]$ "ignore" (diagonal) - Diagonal is ignored "none" - the matrix is used "as is"
use	For use with methods "cor" and "cov", for other methods (the default option should be used if <code>handle.interaction=="ignore"</code> ), "pairwise.complete.obs" are always used, if <code>stats.dist.cor.cov=TRUE</code>
...	Additional arguments to fun

### Details

If both `method` and `fun` are "default", the euclidian distances are computed. the "default" method for `fun="dist"` is "euclidian" and for `fun="cor"` "pearson".

### Value

A matrix (usually of class `dist`) is returned.

**Author(s)**

Aleš Žiberna

**References**

Batagelj, V., Ferligoj, A., Doreian, P. (1992): Direct and indirect methods for structural equivalence. *Social Networks* 14, 63-90.

Lorrain, F., White, H.C., 1971. Structural equivalence of individuals in social networks. *Journal of Mathematical Sociology* 1, 49-80.

**See Also**

[dist](#), [hclust](#), [REGE](#), [crit.fun](#), [opt.par](#), [opt.random.par](#)

**Examples**

```
#generating a simple network corresponding to the simple Sum of squares
#structural equivalence with blockmodel:
# null com
# null null
n<-20
net<-matrix(NA,ncol=n,nrow=n)
clu<-rep(1:2,times=c(5,15))
tclu<-table(clu)
net[clu==1,clu==1]<-rnorm(n=tclu[1]*tclu[1],mean=0,sd=1)
net[clu==1,clu==2]<-rnorm(n=tclu[1]*tclu[2],mean=4,sd=1)
net[clu==2,clu==1]<-rnorm(n=tclu[2]*tclu[1],mean=0,sd=1)
net[clu==2,clu==2]<-rnorm(n=tclu[2]*tclu[2],mean=0,sd=1)

D<-sedist(M=net)
plot.mat(net,clu=cutree(hclust(d=D,method="ward"),k=2))
```

---

 ss

*Sum of Squared deviations from the mean and sum of Absolute Deviations from the median*

---

**Description**

Functions to compute Sum of Squared deviations from the mean and sum of Absolute Deviations from the median

**Usage**

```
ss(x)
ad(x)
```

**Arguments**

x                    A numeric vector.

**Value**

Sum of Squared deviations from the mean or sum of Absolute Deviations from the median

**Author(s)**

Aleš Žiberna

---

two2one

*Two-mode network conversions*

---

**Description**

Covertng two mode networks from two to one mode matrix representation and vice versa. If a two-mode matrix is converted in-to a one-mode matrix, the original two-mode matrix lies in the upper right corner of the one-mode matrix.

**Usage**

```
two2one(M, clu = NULL)
```

```
one2two(M, clu = NULL)
```

**Arguments**

**M** A matrix representing the (usually valued) network.

**clu** A partition. Each unique value represents one cluster. This should be a list of two vectors, one for each mode.

**Value**

Functions returns list with elemets: a mode mode matrix with the two mode network in its upper left corner.

**M** The matrix

**clu** The partition, in form appropriate for the mode of the matrix

**Author(s)**

Aleš Žiberna

**See Also**

[crit.fun](#), [opt.par](#), [opt.random.par](#), [plot.mat](#)

**Examples**

```

#generating a simple network corresponding to the simple Sum of squares
#structural equivalence with blockmodel:
# null com
# null null
n<-c(7,13)
net<-matrix(NA,nrow=n[1],ncol=n[2])
clu<-list(rep(1:2,times=c(3,4)),rep(1:2,times=c(5,8)))
tclu<-lapply(clu,table)
net[clu[[1]]==1,clu[[2]]==1]<-rnorm(n=tclu[[1]][1]*tclu[[2]][1],
  mean=0,sd=1)
net[clu[[1]]==1,clu[[2]]==2]<-rnorm(n=tclu[[1]][1]*tclu[[2]][2],
  mean=4,sd=1)
net[clu[[1]]==2,clu[[2]]==1]<-rnorm(n=tclu[[1]][2]*tclu[[2]][1],
  mean=4,sd=1)
net[clu[[1]]==2,clu[[2]]==2]<-rnorm(n=tclu[[1]][2]*tclu[[2]][2],
  mean=0,sd=1)
plot.mat(net,clu=clu) #two mode matrix of a two mode network
#converting to one mode network
M1<-two2one(net)$M
plot.mat(M1,clu=two2one(net)$clu) #plotting one mode matrix
plot.mat(one2two(M1,clu=clu)$M,clu=clu)
#converting one to two mode matix and plotting

```

# Index

- \*Topic **character**
  - formatA, 14
- \*Topic **cluster**
  - blockmodeling-package, 2
  - check.these.par, 4
  - crit.fun, 7
  - find.m, 12
  - fun.by.blocks, 15
  - genRandomPar, 16
  - nkpartitions, 20
  - opt.par, 21
  - opt.random.par, 24
  - rand, 35
  - REGE, 36
  - sedist, 40
  - two2one, 42
- \*Topic **file**
  - Pajek, 28
- \*Topic **graphs**
  - blockmodeling-package, 2
  - check.these.par, 4
  - crit.fun, 7
  - gplot1, 18
  - opt.par, 21
  - opt.random.par, 24
  - Pajek, 28
  - plot.mat, 30
  - REGE, 36
  - sedist, 40
  - two2one, 42
- \*Topic **hplot**
  - plot.mat, 30
- \*Topic **manip**
  - clu, 6
  - ircNorm, 19
  - recode, 36
  - reorderImage, 39
- \*Topic **math**
  - fun.by.blocks, 15
- \*Topic **package**
  - blockmodeling-package, 2
- \*Topic **univar**
  - ss, 41
- ad (ss), 41
- blockmodeling-package, 2
- check.these.par, 3, 4, 6, 11, 23, 28, 30, 34
- clu, 6, 39
- crand (rand), 35
- crand2 (rand), 35
- crit.fun, 3–6, 7, 14, 21–25, 28, 30, 34, 38, 39, 41, 42
- dist, 41
- err, 39
- err (clu), 6
- find.cut, 14
- find.cut (find.m), 12
- find.m, 12, 14
- find.m2, 14
- find.m2 (find.m), 12
- formatA, 14
- fun.by.blocks, 15
- genRandomPar, 16, 26, 27
- gplot1, 18
- gplot2 (gplot1), 18
- hclust, 41
- IM, 39
- IM (clu), 6
- ircNorm, 19
- loadmatrix (Pajek), 28
- loadnetwork (Pajek), 28

loadnetwork2 (Pajek), 28  
loadpajek (Pajek), 28  
loadvector (Pajek), 28  
loadvector2 (Pajek), 28

network, 3  
nkpar (nkpartitions), 20  
nkpartitions, 5, 20

one2two (two2one), 42  
opt.par, 3, 5, 6, 11, 14, 21, 28, 30, 34, 38, 41, 42  
opt.random.par, 3, 6, 11, 16, 17, 23, 24, 30, 34, 41, 42  
opt.these.par, 3, 5, 6, 11, 16, 23, 30, 34  
opt.these.par (opt.random.par), 24

Pajek, 28  
partitions (clu), 6  
plot, 18  
plot.check.these.par, 5  
plot.check.these.par (plot.mat), 30  
plot.crit.fun, 11  
plot.crit.fun (plot.mat), 30  
plot.mat, 3, 14, 30, 30, 38, 39, 42  
plot.opt.more.par, 28  
plot.opt.more.par (plot.mat), 30  
plot.opt.par, 6, 23  
plot.opt.par (plot.mat), 30

rand, 35  
rand2 (rand), 35  
recode, 36  
REGD.for (REGE), 36  
REGD.ne.for (REGE), 36  
REGD.ow.for (REGE), 36  
REGD.ow.ne.for (REGE), 36  
REGE, 3, 36, 41  
reorderImage, 39

savecluster (Pajek), 28  
savematrix (Pajek), 28  
savenetwork (Pajek), 28  
savepajek (Pajek), 28  
savevector (Pajek), 28  
sedist, 38, 40  
sna, 3  
ss, 41

two2one, 42