# Package 'gRbase'

July 2, 2014

**Version** 1.7-0.1

**Title** A package for graphical modelling in R

**Author** Søren Højsgaard <sorenh@math.aau.dk> with contributions from Claus Dethlefsen <cld@rn.dk> and Clive Bowsher <C.Bowsher@statslab.cam.ac.uk>

**Maintainer** Søren Højsgaard <sorenh@math.aau.dk>

**Description** The gRbase package provides certain general constructs which are used by other graphical modelling packages, in particular by the packages gRain, gRim and gRc.

gRbase contains several datasets relevant for use in connection with graphical models. Almost all datasets used in the book Graphical Models with R (2012) are contained in gRbase.

gRbase implements several graph algorithms (based mainly on representing graphs as adjacency matrices - either in the form of a standard matrix or a sparse matrix). Some graph algorithms are: (i) maximum cardinality search (for marked and unmarked graphs). (ii) moralize. (iii) triangulate. (iv) junctionTree.

gRbase facilities for array operations,

gRbase implements functions for testing for conditional independence.

gRbase illustrates how hierarchical log-linear models (hllm) may be implemented and describes concept of gmData (graphical meta data). These features, however, are not maintained anymore and remains in gRbase only because there exists a paper describing these facilities: A Common Platform for Graphical Models in R: The gRbase Package, Journal of Statistical Software, Vol 14,No 17, 2005.

**License** GPL (>= 2)

**URL** http://people.math.aau.dk/~sorenh/software/gR/

**ByteCompile** Yes

**Encoding** latin1

**Depends** R (>= 3.0.2), methods

**Imports** Matrix, RBGL, igraph, graph, Rcpp (>= 0.11.1)

**Suggests** Rgraphviz, microbenchmark

**LinkingTo** Rcpp (>= 0.11.1), RcppArmadillo, RcppEigen

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2014-03-23 07:04:39

# R topics documented:

---

`arrayCombine` *Combine arrays*

---

## Description

Combine arrays into a new array with higher dimension.

## Usage

```
arrayCombine(aa.list, aux)
arrayExtendDomain(aa, bb)
```

## Arguments

| | |
|---|---|
| `aa.list` | List of arrays. |
| `aux` | A list with one element. The element must be a vector and the element must be named, e.g. `list(Z=c(1,2))`. |
| `aa` | An array |
| `bb` | A list with additional dimensions to be added, e.g. `list(Z=c(1,2), U=c("a","b"))` |

## Value

An array

## Note

For two arrays with a common variable, it is not checked that the levels of that variable match. They must match, but it is the users responsibility to check that they do.

## Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

## Examples

```
## Case 1: t1 and t2 are arrays defined over identical sets of variables:
t1 <- parray(c("y","x1"), c(2,2), 1:4)
t2 <- parray(c("y","x1"), c(2,2), c(-11,12,-13,14))
tc <- arrayCombine(list(t1,t2), aux=list(Z=c(1,2)))
as.data.frame.table(tc)
## The "auxilary" variable Z adds a new dimension to the table

## Case 2: t1 and t2 are arrays defined over non-identical sets of variables:
t1 <- parray(c("y","x1"), c(2,2), 1:4)
t2 <- parray(c("y","x2"), c(2,2), c(-11,12,-13,14))
tc <- arrayCombine(list(t1,t2), aux=list(Z=c(1,2)))
as.data.frame.table(tc)
## The "auxilary" variable Z adds a new dimension to the table
## When Z=Z1, tc is constant as a function of x2
## When Z=Z2, tc is constant as a function of x1

## Case 3: t1 and t2 are arrays defined over non-identical sets of variables,
## but the variables for t1 is a subset of the variables for t2:
t1 <- parray(c("y","x1"), c(2,2), 1:4)
t2 <- parray(c("y","x1","x2"), c(2,2,2), 11:18)
tc <- arrayCombine(list(t1,t2), aux=list(Z=c(1,2)))
as.data.frame.table(tc)
## The "auxilary" variable Z adds a new dimension to the table
## When Z=Z1, tc is constant as a function of x2
```

---

# ashtrees *Crown dieback in ash trees*

---

## Description

This dataset comes from a study of symptoms of crown dieback, cankers and symptoms caused by other pathogens and pests in ash trees (Fraxinus excelsior). In all 454 trees were observed in two plots. There are 8 categorical variables, 6 of which are binary and two are trichotomous with values representing increasing severity of symptoms, and one continuous variable, tree diameter at breast height (DBH).

## Usage

```
data(ashtrees)
```

## Format

A data frame with 454 observations on the following 9 variables.

`plot` a factor with levels `2` `6`

`dieback` a factor with levels `0` `1` `2`

`dead50` a factor with levels `0` `0.5` `1`

`bushy` a factor with levels `0` `1`

`canker` a factor with levels `BRNCH` `MAIN` `NONE`

`wilt` a factor with levels `0` `1`

`roses` a factor with levels `0` `1`

`discolour` a factor with levels `0` `1`

`dbh` a numeric vector

## References

Skovgaard JP, Thomsen IM, Skovgaard IM and Martinussen T (2009). Associations among symptoms of dieback in even-aged stands of ash (Fraxinus excelsior L.). Forest Pathology.

## Examples

```
data(ashtrees)
## maybe str(ashtrees) ; plot(ashtrees) ...
```

---

`BodyFat` *Body Fat Data*

---

## Description

Estimates of the percentage of body fat determined by underwater weighing and various body circumference measurements for 252 men.

## Usage

```
data(BodyFat)
```

## Format

A data frame with 252 observations on the following 15 variables.

`Density` Density determined from underwater weighing, a numeric vector

`BodyFat` Percent body fat from Siri's (1956) equation, a numeric vector

`Age` in years, a numeric vector

`Weight` in lbs, a numeric vector

`Height` in inches, a numeric vector

`Neck` circumference in cm, a numeric vector

`Chest` circumference in cm, a numeric vector

`Abdomen` circumference in cm, a numeric vector

`Hip` circumference in cm, a numeric vector

`Thigh` circumference in cm, a numeric vector

`Knee` circumference in cm, a numeric vector

`Ankle` circumference in cm, a numeric vector

`Biceps` circumference in cm, a numeric vector

`Forearm` circumference in cm, a numeric vector

`Wrist` circumference in cm, a numeric vector

## Source

For more information see http://lib.stat.cmu.edu/datasets/bodyfat

## References

Bailey, Covert (1994). _Smart Exercise: Burning Fat, Getting Fit_, Houghton-Mifflin Co., Boston, pp. 179-186.

Behnke, A.R. and Wilmore, J.H. (1974). _Evaluation and Regulation of Body Build and Composition_, Prentice-Hall, Englewood Cliffs, N.J.

Siri, W.E. (1956), "Gross composition of the body", in _Advances in Biological and Medical Physics_, vol. IV, edited by J.H. Lawrence and C.A. Tobias, Academic Press, Inc., New York.

Katch, Frank and McArdle, William (1977). _Nutrition, Weight Control, and Exercise_, Houghton Mifflin Co., Boston.

Wilmore, Jack (1976). _Athletic Training and Physical Fitness: Physiological Principles of the Conditioning Process_, Allyn and Bacon, Inc., Boston.

## Examples

```
data(BodyFat)
head(BodyFat)
```

---

# breastcancer *Gene expression signatures for p53 mutation status in 250 breast cancer samples*

---

## Description

Perturbations of the p53 pathway are associated with more aggressive and therapeutically refractory tumours. We preprocessed the data using Robust Multichip Analysis (RMA). Dataset has been truncated to the 1000 most informative genes (as selected by Wilcoxon test statistics) to simplify computation. The genes have been standardised to have zero mean and unit variance (i.e. z-scored).

## Usage

```
data(breastcancer)
```

## Format

A data frame with 250 observations on the following 1001 variables.

`A.1053_at` a numeric vector

`A.200039_s_at` a numeric vector

`A.200053_at` a numeric vector

`A.200079_s_at` a numeric vector

`A.200628_s_at` a numeric vector

`A.200639_s_at` a numeric vector

`A.200670_at` a numeric vector

`A.200687_s_at` a numeric vector

`A.200710_at` a numeric vector

`A.200740_s_at` a numeric vector

`A.200773_x_at` a numeric vector

`A.200783_s_at` a numeric vector

`A.200795_at` a numeric vector

`A.200810_s_at` a numeric vector

`A.200811_at` a numeric vector

`A.200822_x_at` a numeric vector

`A.200840_at` a numeric vector

`A.200853_at` a numeric vector

`A.200854_at` a numeric vector

`A.200855_at` a numeric vector

`A.200996_at` a numeric vector

`A.201041_s_at` a numeric vector

`A.201077_s_at` a numeric vector

`A.201088_at` a numeric vector

`A.201104_x_at` a numeric vector

`A.201114_x_at` a numeric vector

`A.201115_at` a numeric vector

`A.201124_at` a numeric vector

`A.201140_s_at` a numeric vector

`A.201170_s_at` a numeric vector

`A.201195_s_at` a numeric vector

`A.201196_s_at` a numeric vector

`A.201197_at` a numeric vector

`A.201201_at` a numeric vector

`A.201202_at` a numeric vector

A.201236_s_at a numeric vector

A.201263_at a numeric vector

A.201266_at a numeric vector

A.201281_at a numeric vector

A.201291_s_at a numeric vector

A.201292_at a numeric vector

A.201327_s_at a numeric vector

A.201342_at a numeric vector

A.201384_s_at a numeric vector

A.201394_s_at a numeric vector

A.201395_at a numeric vector

A.201397_at a numeric vector

A.201455_s_at a numeric vector

A.201479_at a numeric vector

A.201508_at a numeric vector

A.201557_at a numeric vector

A.201584_s_at a numeric vector

A.201591_s_at a numeric vector

A.201663_s_at a numeric vector

A.201664_at a numeric vector

A.201685_s_at a numeric vector

A.201694_s_at a numeric vector

A.201710_at a numeric vector

A.201755_at a numeric vector

A.201761_at a numeric vector

A.201770_at a numeric vector

A.201790_s_at a numeric vector

A.201791_s_at a numeric vector

A.201795_at a numeric vector

A.201833_at a numeric vector

A.201890_at a numeric vector

A.201896_s_at a numeric vector

A.201897_s_at a numeric vector

A.201930_at a numeric vector

A.201970_s_at a numeric vector

A.201977_s_at a numeric vector

A.201978_s_at a numeric vector

A.202011_at a numeric vector

A.202088_at a numeric vector

A.202095_s_at a numeric vector

A.202105_at a numeric vector

A.202107_s_at a numeric vector

A.202109_at a numeric vector

A.202117_at a numeric vector

A.202174_s_at a numeric vector

A.202188_at a numeric vector

A.202200_s_at a numeric vector

A.202233_s_at a numeric vector

A.202240_at a numeric vector

A.202270_at a numeric vector

A.202276_at a numeric vector

A.202307_s_at a numeric vector

A.202338_at a numeric vector

A.202370_s_at a numeric vector

A.202371_at a numeric vector

A.202409_at a numeric vector

A.202487_s_at a numeric vector

A.202503_s_at a numeric vector

A.202580_x_at a numeric vector

A.202589_at a numeric vector

A.202590_s_at a numeric vector

A.202613_at a numeric vector

A.202666_s_at a numeric vector

A.202690_s_at a numeric vector

A.202705_at a numeric vector

A.202725_at a numeric vector

A.202754_at a numeric vector

A.202779_s_at a numeric vector

A.202808_at a numeric vector

A.202815_s_at a numeric vector

A.202854_at a numeric vector

A.202858_at a numeric vector

A.202870_s_at a numeric vector

A.202954_at a numeric vector

A.202962_at a numeric vector

A.203022_at a numeric vector

A.203046_s_at a numeric vector

A.203066_at a numeric vector

A.203071_at a numeric vector

A.203126_at a numeric vector

A.203130_s_at a numeric vector

A.203139_at a numeric vector

A.203145_at a numeric vector

A.203187_at a numeric vector

A.203188_at a numeric vector

A.203208_s_at a numeric vector

A.203213_at a numeric vector

A.203214_x_at a numeric vector

A.203223_at a numeric vector

A.203249_at a numeric vector

A.203265_s_at a numeric vector

A.203266_s_at a numeric vector

A.203276_at a numeric vector

A.203287_at a numeric vector

A.203296_s_at a numeric vector

A.203345_s_at a numeric vector

A.203347_s_at a numeric vector

A.203358_s_at a numeric vector

A.203362_s_at a numeric vector

A.203380_x_at a numeric vector

A.203405_at a numeric vector

A.203418_at a numeric vector

A.203422_at a numeric vector

A.203428_s_at a numeric vector

A.203432_at a numeric vector

A.203438_at a numeric vector

A.203439_s_at a numeric vector

A.203463_s_at a numeric vector

A.203484_at a numeric vector

A.203554_x_at a numeric vector

A.203560_at a numeric vector

A.203594_at a numeric vector

A.203625_x_at a numeric vector

A.203693_s_at a numeric vector

A.203696_s_at a numeric vector

A.203702_s_at a numeric vector

A.203744_at a numeric vector

A.203755_at a numeric vector

A.203764_at a numeric vector

A.203799_at a numeric vector

A.203801_at a numeric vector

A.203805_s_at a numeric vector

A.203806_s_at a numeric vector

A.203856_at a numeric vector

A.203859_s_at a numeric vector

A.203906_at a numeric vector

A.203929_s_at a numeric vector

A.203930_s_at a numeric vector

A.203967_at a numeric vector

A.203968_s_at a numeric vector

A.204020_at a numeric vector

A.204023_at a numeric vector

A.204026_s_at a numeric vector

A.204033_at a numeric vector

A.204045_at a numeric vector

A.204072_s_at a numeric vector

A.204092_s_at a numeric vector

A.204115_at a numeric vector

A.204126_s_at a numeric vector

A.204127_at a numeric vector

A.204146_at a numeric vector

A.204162_at a numeric vector

A.204170_s_at a numeric vector

A.204203_at a numeric vector

A.204217_s_at a numeric vector

A.204244_s_at a numeric vector

A.204267_x_at a numeric vector

A.204290_s_at a numeric vector

A.204315_s_at a numeric vector

A.204317_at a numeric vector

A.204318_s_at a numeric vector

A.204441_s_at a numeric vector

A.204444_at a numeric vector

A.204482_at a numeric vector

A.204510_at a numeric vector

A.204533_at a numeric vector

A.204558_at a numeric vector

A.204592_at a numeric vector

A.204603_at a numeric vector

A.204640_s_at a numeric vector

A.204641_at a numeric vector

A.204649_at a numeric vector

A.204675_at a numeric vector

A.204686_at a numeric vector

A.204695_at a numeric vector

A.204702_s_at a numeric vector

A.204709_s_at a numeric vector

A.204732_s_at a numeric vector

A.204766_s_at a numeric vector

A.204767_s_at a numeric vector

A.204768_s_at a numeric vector

A.204800_s_at a numeric vector

A.204817_at a numeric vector

A.204822_at a numeric vector

A.204825_at a numeric vector

A.204863_s_at a numeric vector

A.204881_s_at a numeric vector

A.204962_s_at a numeric vector

A.205024_s_at a numeric vector

A.205034_at a numeric vector

A.205046_at a numeric vector

A.205074_at a numeric vector

A.205158_at a numeric vector

A.205167_s_at a numeric vector

A.205186_at a numeric vector

A.205214_at a numeric vector

A.205225_at a numeric vector

A.205240_at a numeric vector

A.205279_s_at a numeric vector

A.205280_at a numeric vector

A.205282_at a numeric vector

A.205296_at a numeric vector

A.205339_at a numeric vector

A.205354_at a numeric vector

A.205392_s_at a numeric vector

A.205393_s_at a numeric vector

A.205394_at a numeric vector

A.205436_s_at a numeric vector

A.205440_s_at a numeric vector

A.205471_s_at a numeric vector

A.205472_s_at a numeric vector

A.205569_at a numeric vector

A.205611_at a numeric vector

A.205628_at a numeric vector

A.205644_s_at a numeric vector

A.205672_at a numeric vector

A.205710_at a numeric vector

A.205711_x_at a numeric vector

A.205733_at a numeric vector

A.205776_at a numeric vector

A.205779_at a numeric vector

A.205794_s_at a numeric vector

A.205883_at a numeric vector

A.205898_at a numeric vector

A.205933_at a numeric vector

A.205943_at a numeric vector

A.205967_at a numeric vector

A.206055_s_at a numeric vector

A.206074_s_at a numeric vector

A.206091_at a numeric vector

A.206102_at a numeric vector

A.206134_at a numeric vector

A.206197_at a numeric vector

A.206245_s_at a numeric vector

A.206364_at a numeric vector

A.206481_s_at a numeric vector

A.206499_s_at a numeric vector

A.206600_s_at a numeric vector

A.206632_s_at a numeric vector

A.206702_at a numeric vector

A.206794_at a numeric vector

A.206869_at a numeric vector

A.207165_at a numeric vector

A.207542_s_at a numeric vector

A.207783_x_at a numeric vector

A.207788_s_at a numeric vector

A.207828_s_at a numeric vector

A.208029_s_at a numeric vector

A.208079_s_at a numeric vector

A.208103_s_at a numeric vector

A.208140_s_at a numeric vector

A.208184_s_at a numeric vector

A.208190_s_at a numeric vector

A.208308_s_at a numeric vector

A.208374_s_at a numeric vector

A.208433_s_at a numeric vector

A.208503_s_at a numeric vector

A.208511_at a numeric vector

A.208517_x_at a numeric vector

A.208614_s_at a numeric vector

A.208627_s_at a numeric vector

A.208628_s_at a numeric vector

A.208682_s_at a numeric vector

A.208691_at a numeric vector

A.208693_s_at a numeric vector

A.208696_at a numeric vector

A.208709_s_at a numeric vector

A.208718_at a numeric vector

A.208767_s_at a numeric vector

A.208794_s_at a numeric vector

A.208804_s_at a numeric vector

A.208807_s_at a numeric vector

A.208870_x_at a numeric vector

A.208905_at a numeric vector

A.208952_s_at a numeric vector

A.208969_at a numeric vector

A.209002_s_at a numeric vector

A.209019_s_at a numeric vector

A.209026_x_at a numeric vector

A.209068_at a numeric vector

A.209070_s_at a numeric vector

A.209071_s_at a numeric vector

A.209075_s_at a numeric vector

A.209172_s_at a numeric vector

A.209189_at a numeric vector

A.209195_s_at a numeric vector

A.209218_at a numeric vector

A.209311_at a numeric vector

A.209336_at a numeric vector

A.209341_s_at a numeric vector

A.209358_at a numeric vector

A.209375_at a numeric vector

A.209408_at a numeric vector

A.209459_s_at a numeric vector

A.209460_at a numeric vector

A.209464_at a numeric vector

A.209482_at a numeric vector

A.209523_at a numeric vector

A.209603_at a numeric vector

A.209604_s_at a numeric vector

A.209608_s_at a numeric vector

A.209623_at a numeric vector

A.209625_at a numeric vector

A.209642_at a numeric vector

A.209680_s_at a numeric vector

A.209709_s_at a numeric vector

A.209714_s_at a numeric vector

A.209737_at a numeric vector

A.209740_s_at a numeric vector

A.209747_at a numeric vector

A.209753_s_at a numeric vector

A.209773_s_at a numeric vector

A.209825_s_at a numeric vector

A.209832_s_at a numeric vector

A.209869_at a numeric vector

A.209891_at a numeric vector

A.209897_s_at a numeric vector

A.210052_s_at a numeric vector

A.210092_at a numeric vector

A.210093_s_at a numeric vector

A.210108_at a numeric vector

A.210137_s_at a numeric vector

A.210163_at a numeric vector

A.210346_s_at a numeric vector

A.210365_at a numeric vector

A.210416_s_at a numeric vector

A.210466_s_at a numeric vector

A.210559_s_at a numeric vector

A.210567_s_at a numeric vector

A.210766_s_at a numeric vector

A.210778_s_at a numeric vector

A.210821_x_at a numeric vector

A.210881_s_at a numeric vector

A.211000_s_at a numeric vector

A.211034_s_at a numeric vector

A.211042_x_at a numeric vector

A.211072_x_at a numeric vector

A.211080_s_at a numeric vector

A.211378_x_at a numeric vector

A.211519_s_at a numeric vector

A.211714_x_at a numeric vector

A.211725_s_at a numeric vector

A.211765_x_at a numeric vector

A.211939_x_at a numeric vector

A.211943_x_at a numeric vector

A.211967_at a numeric vector

A.211986_at a numeric vector

A.212020_s_at a numeric vector

A.212022_s_at a numeric vector

A.212023_s_at a numeric vector

A.212048_s_at a numeric vector

A.212151_at a numeric vector

A.212195_at a numeric vector

A.212196_at a numeric vector

A.212209_at a numeric vector

A.212219_at a numeric vector

A.212228_s_at a numeric vector

A.212242_at a numeric vector

A.212247_at a numeric vector

A.212266_s_at a numeric vector

A.212284_x_at a numeric vector

A.212296_at a numeric vector

A.212299_at a numeric vector

A.212319_at a numeric vector

A.212419_at a numeric vector

A.212423_at a numeric vector

A.212430_at a numeric vector

A.212448_at a numeric vector

A.212492_s_at a numeric vector

A.212494_at a numeric vector

A.212495_at a numeric vector

A.212496_s_at a numeric vector

A.212541_at a numeric vector

A.212581_x_at a numeric vector

A.212660_at a numeric vector

A.212680_x_at a numeric vector

A.212695_at a numeric vector

A.212705_x_at a numeric vector

A.212708_at a numeric vector

A.212744_at a numeric vector

A.212745_s_at a numeric vector

A.212779_at a numeric vector

A.212815_at a numeric vector

A.212846_at a numeric vector

A.212865_s_at a numeric vector

A.212869_x_at a numeric vector

A.212914_at a numeric vector

A.212936_at a numeric vector

A.212948_at a numeric vector

A.212949_at a numeric vector

A.212956_at a numeric vector

A.212970_at a numeric vector

A.212978_at a numeric vector

A.212985_at a numeric vector

A.213007_at a numeric vector

A.213008_at a numeric vector

A.213011_s_at a numeric vector

A.213018_at a numeric vector

A.213058_at a numeric vector

A.213063_at a numeric vector

A.213088_s_at a numeric vector

A.213101_s_at a numeric vector

A.213102_at a numeric vector

A.213103_at a numeric vector

A.213137_s_at a numeric vector

A.213175_s_at a numeric vector

A.213195_at a numeric vector

A.213224_s_at a numeric vector

A.213226_at a numeric vector

A.213283_s_at a numeric vector

A.213285_at a numeric vector

A.213366_x_at a numeric vector

A.213397_x_at a numeric vector

A.213427_at a numeric vector

A.213451_x_at a numeric vector

A.213453_x_at a numeric vector

A.213464_at a numeric vector

A.213519_s_at a numeric vector

A.213520_at a numeric vector

A.213523_at a numeric vector

A.213541_s_at a numeric vector

A.213571_s_at a numeric vector

A.213577_at a numeric vector

A.213590_at a numeric vector

A.213599_at a numeric vector

A.213627_at a numeric vector

A.213671_s_at a numeric vector

A.213702_x_at a numeric vector

A.213720_s_at a numeric vector

A.213748_at a numeric vector

A.213832_at a numeric vector

A.213911_s_at a numeric vector

A.213933_at a numeric vector

A.214039_s_at a numeric vector

A.214053_at a numeric vector

A.214061_at a numeric vector

A.214062_x_at a numeric vector

A.214077_x_at a numeric vector

A.214095_at a numeric vector

A.214096_s_at a numeric vector

A.214101_s_at a numeric vector

A.214144_at a numeric vector

A.214164_x_at a numeric vector

A.214188_at a numeric vector

A.214269_at a numeric vector

A.214431_at a numeric vector

A.214435_x_at a numeric vector

A.214437_s_at a numeric vector

A.214440_at a numeric vector

A.214552_s_at a numeric vector

A.214581_x_at a numeric vector

A.214657_s_at a numeric vector

A.214710_s_at a numeric vector

A.214728_x_at a numeric vector

A.214745_at a numeric vector
A.214919_s_at a numeric vector
A.215014_at a numeric vector
A.215146_s_at a numeric vector
A.215223_s_at a numeric vector
A.215300_s_at a numeric vector
A.215303_at a numeric vector
A.215304_at a numeric vector
A.215616_s_at a numeric vector
A.215722_s_at a numeric vector
A.215867_x_at a numeric vector
A.215942_s_at a numeric vector
A.216000_at a numeric vector
A.216088_s_at a numeric vector
A.216109_at a numeric vector
A.216237_s_at a numeric vector
A.216264_s_at a numeric vector
A.216465_at a numeric vector
A.216515_x_at a numeric vector
A.216520_s_at a numeric vector
A.216841_s_at a numeric vector
A.216885_s_at a numeric vector
A.216952_s_at a numeric vector
A.216977_x_at a numeric vector
A.217297_s_at a numeric vector
A.217346_at a numeric vector
A.217398_x_at a numeric vector
A.217427_s_at a numeric vector
A.217499_x_at a numeric vector
A.217714_x_at a numeric vector
A.217733_s_at a numeric vector
A.217755_at a numeric vector
A.217834_s_at a numeric vector
A.217838_s_at a numeric vector
A.217844_at a numeric vector
A.217889_s_at a numeric vector
A.217906_at a numeric vector

A.217925_s_at a numeric vector

A.217938_s_at a numeric vector

A.217990_at a numeric vector

A.218007_s_at a numeric vector

A.218009_s_at a numeric vector

A.218039_at a numeric vector

A.218065_s_at a numeric vector

A.218074_at a numeric vector

A.218080_x_at a numeric vector

A.218115_at a numeric vector

A.218131_s_at a numeric vector

A.218146_at a numeric vector

A.218193_s_at a numeric vector

A.218211_s_at a numeric vector

A.218213_s_at a numeric vector

A.218219_s_at a numeric vector

A.218239_s_at a numeric vector

A.218259_at a numeric vector

A.218260_at a numeric vector

A.218295_s_at a numeric vector

A.218308_at a numeric vector

A.218349_s_at a numeric vector

A.218350_s_at a numeric vector

A.218353_at a numeric vector

A.218355_at a numeric vector

A.218374_s_at a numeric vector

A.218384_at a numeric vector

A.218418_s_at a numeric vector

A.218437_s_at a numeric vector

A.218447_at a numeric vector

A.218471_s_at a numeric vector

A.218474_s_at a numeric vector

A.218483_s_at a numeric vector

A.218497_s_at a numeric vector

A.218499_at a numeric vector

A.218512_at a numeric vector

A.218538_s_at a numeric vector

A.218542_at a numeric vector
A.218552_at a numeric vector
A.218584_at a numeric vector
A.218585_s_at a numeric vector
A.218588_s_at a numeric vector
A.218662_s_at a numeric vector
A.218663_at a numeric vector
A.218684_at a numeric vector
A.218726_at a numeric vector
A.218728_s_at a numeric vector
A.218741_at a numeric vector
A.218755_at a numeric vector
A.218769_s_at a numeric vector
A.218795_at a numeric vector
A.218834_s_at a numeric vector
A.218844_at a numeric vector
A.218856_at a numeric vector
A.218875_s_at a numeric vector
A.218877_s_at a numeric vector
A.218883_s_at a numeric vector
A.218894_s_at a numeric vector
A.218936_s_at a numeric vector
A.218966_at a numeric vector
A.218976_at a numeric vector
A.218982_s_at a numeric vector
A.218987_at a numeric vector
A.219000_s_at a numeric vector
A.219004_s_at a numeric vector
A.219010_at a numeric vector
A.219031_s_at a numeric vector
A.219061_s_at a numeric vector
A.219065_s_at a numeric vector
A.219091_s_at a numeric vector
A.219105_x_at a numeric vector
A.219132_at a numeric vector
A.219148_at a numeric vector
A.219197_s_at a numeric vector

A.219252_s_at a numeric vector

A.219286_s_at a numeric vector

A.219304_s_at a numeric vector

A.219306_at a numeric vector

A.219396_s_at a numeric vector

A.219413_at a numeric vector

A.219417_s_at a numeric vector

A.219436_s_at a numeric vector

A.219440_at a numeric vector

A.219455_at a numeric vector

A.219463_at a numeric vector

A.219469_at a numeric vector

A.219490_s_at a numeric vector

A.219493_at a numeric vector

A.219494_at a numeric vector

A.219510_at a numeric vector

A.219555_s_at a numeric vector

A.219588_s_at a numeric vector

A.219646_at a numeric vector

A.219650_at a numeric vector

A.219686_at a numeric vector

A.219687_at a numeric vector

A.219689_at a numeric vector

A.219713_at a numeric vector

A.219787_s_at a numeric vector

A.219833_s_at a numeric vector

A.219918_s_at a numeric vector

A.219922_s_at a numeric vector

A.219978_s_at a numeric vector

A.219990_at a numeric vector

A.220011_at a numeric vector

A.220060_s_at a numeric vector

A.220094_s_at a numeric vector

A.220173_at a numeric vector

A.220239_at a numeric vector

A.220276_at a numeric vector

A.220324_at a numeric vector

A.220540_at a numeric vector

A.220651_s_at a numeric vector

A.220789_s_at a numeric vector

A.220865_s_at a numeric vector

A.220917_s_at a numeric vector

A.221203_s_at a numeric vector

A.221207_s_at a numeric vector

A.221221_s_at a numeric vector

A.221258_s_at a numeric vector

A.221272_s_at a numeric vector

A.221275_s_at a numeric vector

A.221276_s_at a numeric vector

A.221436_s_at a numeric vector

A.221505_at a numeric vector

A.221520_s_at a numeric vector

A.221521_s_at a numeric vector

A.221562_s_at a numeric vector

A.221588_x_at a numeric vector

A.221590_s_at a numeric vector

A.221676_s_at a numeric vector

A.221677_s_at a numeric vector

A.221685_s_at a numeric vector

A.221740_x_at a numeric vector

A.221856_s_at a numeric vector

A.221913_at a numeric vector

A.221922_at a numeric vector

A.221951_at a numeric vector

A.222010_at a numeric vector

A.222011_s_at a numeric vector

A.222039_at a numeric vector

A.222043_at a numeric vector

A.222077_s_at a numeric vector

A.222118_at a numeric vector

A.222195_s_at a numeric vector

A.36129_at a numeric vector

A.37152_at a numeric vector

A.38158_at a numeric vector

A.39854_r_at a numeric vector

A.40016_g_at a numeric vector

A.40093_at a numeric vector

A.45297_at a numeric vector

A.45633_at a numeric vector

A.57703_at a numeric vector

A.AFFX.HUMGAPDH.M33197_3_at a numeric vector

A.AFFX.HUMGAPDH.M33197_M_at a numeric vector

B.200039_s_at a numeric vector

B.200076_s_at a numeric vector

B.200079_s_at a numeric vector

B.200086_s_at a numeric vector

B.222396_at a numeric vector

B.222423_at a numeric vector

B.222453_at a numeric vector

B.222602_at a numeric vector

B.222606_at a numeric vector

B.222608_s_at a numeric vector

B.222640_at a numeric vector

B.222680_s_at a numeric vector

B.222740_at a numeric vector

B.222752_s_at a numeric vector

B.222767_s_at a numeric vector

B.222820_at a numeric vector

B.222835_at a numeric vector

B.222843_at a numeric vector

B.222848_at a numeric vector

B.222889_at a numeric vector

B.222958_s_at a numeric vector

B.222962_s_at a numeric vector

B.223054_at a numeric vector

B.223056_s_at a numeric vector

B.223062_s_at a numeric vector

B.223068_at a numeric vector

B.223096_at a numeric vector

B.223100_s_at a numeric vector

B.223119_s_at a numeric vector

B.223126_s_at a numeric vector

B.223186_at a numeric vector

B.223204_at a numeric vector

B.223225_s_at a numeric vector

B.223229_at a numeric vector

B.223234_at a numeric vector

B.223274_at a numeric vector

B.223307_at a numeric vector

B.223315_at a numeric vector

B.223348_x_at a numeric vector

B.223361_at a numeric vector

B.223381_at a numeric vector

B.223387_at a numeric vector

B.223452_s_at a numeric vector

B.223480_s_at a numeric vector

B.223491_at a numeric vector

B.223515_s_at a numeric vector

B.223556_at a numeric vector

B.223570_at a numeric vector

B.223666_at a numeric vector

B.223700_at a numeric vector

B.223864_at a numeric vector

B.224217_s_at a numeric vector

B.224320_s_at a numeric vector

B.224333_s_at a numeric vector

B.224428_s_at a numeric vector

B.224468_s_at a numeric vector

B.224471_s_at a numeric vector

B.224521_s_at a numeric vector

B.224523_s_at a numeric vector

B.224566_at a numeric vector

B.224578_at a numeric vector

B.224610_at a numeric vector

B.224674_at a numeric vector

B.224686_x_at a numeric vector

B.224699_s_at a numeric vector

B.224752_at a numeric vector

B.224753_at a numeric vector

B.224903_at a numeric vector

B.224944_at a numeric vector

B.225004_at a numeric vector

B.225064_at a numeric vector

B.225071_at a numeric vector

B.225083_at a numeric vector

B.225092_at a numeric vector

B.225099_at a numeric vector

B.225100_at a numeric vector

B.225142_at a numeric vector

B.225162_at a numeric vector

B.225191_at a numeric vector

B.225268_at a numeric vector

B.225291_at a numeric vector

B.225300_at a numeric vector

B.225309_at a numeric vector

B.225327_at a numeric vector

B.225365_at a numeric vector

B.225379_at a numeric vector

B.225402_at a numeric vector

B.225454_at a numeric vector

B.225468_at a numeric vector

B.225501_at a numeric vector

B.225592_at a numeric vector

B.225611_at a numeric vector

B.225613_at a numeric vector

B.225629_s_at a numeric vector

B.225655_at a numeric vector

B.225656_at a numeric vector

B.225676_s_at a numeric vector

B.225686_at a numeric vector

B.225687_at a numeric vector

B.225723_at a numeric vector

B.225748_at a numeric vector

B.225777_at a numeric vector

B.225804_at a numeric vector

B.225834_at a numeric vector
B.225841_at a numeric vector
B.225866_at a numeric vector
B.226030_at a numeric vector
B.226108_at a numeric vector
B.226118_at a numeric vector
B.226198_at a numeric vector
B.226274_at a numeric vector
B.226298_at a numeric vector
B.226303_at a numeric vector
B.226308_at a numeric vector
B.226344_at a numeric vector
B.226346_at a numeric vector
B.226349_at a numeric vector
B.226358_at a numeric vector
B.226410_at a numeric vector
B.226437_at a numeric vector
B.226439_s_at a numeric vector
B.226456_at a numeric vector
B.226466_s_at a numeric vector
B.226473_at a numeric vector
B.226506_at a numeric vector
B.226519_s_at a numeric vector
B.226522_at a numeric vector
B.226582_at a numeric vector
B.226661_at a numeric vector
B.226831_at a numeric vector
B.226833_at a numeric vector
B.226846_at a numeric vector
B.226914_at a numeric vector
B.226915_s_at a numeric vector
B.226936_at a numeric vector
B.226974_at a numeric vector
B.226977_at a numeric vector
B.226980_at a numeric vector
B.226992_at a numeric vector
B.227021_at a numeric vector

B.227047_x_at a numeric vector

B.227068_at a numeric vector

B.227081_at a numeric vector

B.227103_s_at a numeric vector

B.227165_at a numeric vector

B.227182_at a numeric vector

B.227198_at a numeric vector

B.227211_at a numeric vector

B.227212_s_at a numeric vector

B.227227_at a numeric vector

B.227232_at a numeric vector

B.227279_at a numeric vector

B.227350_at a numeric vector

B.227363_s_at a numeric vector

B.227419_x_at a numeric vector

B.227423_at a numeric vector

B.227436_at a numeric vector

B.227478_at a numeric vector

B.227512_at a numeric vector

B.227533_at a numeric vector

B.227651_at a numeric vector

B.227700_x_at a numeric vector

B.227703_s_at a numeric vector

B.227718_at a numeric vector

B.227762_at a numeric vector

B.227809_at a numeric vector

B.227811_at a numeric vector

B.227873_at a numeric vector

B.227874_at a numeric vector

B.227920_at a numeric vector

B.227928_at a numeric vector

B.227982_at a numeric vector

B.228044_at a numeric vector

B.228069_at a numeric vector

B.228081_at a numeric vector

B.228252_at a numeric vector

B.228273_at a numeric vector

B.228281_at a numeric vector

B.228318_s_at a numeric vector

B.228323_at a numeric vector

B.228327_x_at a numeric vector

B.228361_at a numeric vector

B.228468_at a numeric vector

B.228469_at a numeric vector

B.228476_at a numeric vector

B.228504_at a numeric vector

B.228505_s_at a numeric vector

B.228528_at a numeric vector

B.228550_at a numeric vector

B.228554_at a numeric vector

B.228559_at a numeric vector

B.228560_at a numeric vector

B.228597_at a numeric vector

B.228692_at a numeric vector

B.228718_at a numeric vector

B.228729_at a numeric vector

B.228730_s_at a numeric vector

B.228750_at a numeric vector

B.228799_at a numeric vector

B.228811_at a numeric vector

B.228854_at a numeric vector

B.228868_x_at a numeric vector

B.228915_at a numeric vector

B.228931_at a numeric vector

B.228955_at a numeric vector

B.229030_at a numeric vector

B.229062_at a numeric vector

B.229097_at a numeric vector

B.229127_at a numeric vector

B.229150_at a numeric vector

B.229169_at a numeric vector

B.229170_s_at a numeric vector

B.229181_s_at a numeric vector

B.229342_at a numeric vector

B.229381_at a numeric vector

B.229437_at a numeric vector

B.229466_at a numeric vector

B.229490_s_at a numeric vector

B.229538_s_at a numeric vector

B.229551_x_at a numeric vector

B.229610_at a numeric vector

B.229764_at a numeric vector

B.229886_at a numeric vector

B.230021_at a numeric vector

B.230123_at a numeric vector

B.230142_s_at a numeric vector

B.230165_at a numeric vector

B.230250_at a numeric vector

B.230451_at a numeric vector

B.230469_at a numeric vector

B.230863_at a numeric vector

B.230966_at a numeric vector

B.231002_s_at a numeric vector

B.231034_s_at a numeric vector

B.231472_at a numeric vector

B.231577_s_at a numeric vector

B.232065_x_at a numeric vector

B.232210_at a numeric vector

B.232238_at a numeric vector

B.232278_s_at a numeric vector

B.232286_at a numeric vector

B.232307_at a numeric vector

B.232398_at a numeric vector

B.232459_at a numeric vector

B.232596_at a numeric vector

B.232855_at a numeric vector

B.232944_at a numeric vector

B.232968_at a numeric vector

B.233110_s_at a numeric vector

B.233388_at a numeric vector

B.233413_at a numeric vector

B.233520_s_at a numeric vector

B.234222_at a numeric vector

B.234294_x_at a numeric vector

B.234749_s_at a numeric vector

B.234863_x_at a numeric vector

B.234944_s_at a numeric vector

B.234954_at a numeric vector

B.234992_x_at a numeric vector

B.235040_at a numeric vector

B.235046_at a numeric vector

B.235117_at a numeric vector

B.235124_at a numeric vector

B.235181_at a numeric vector

B.235193_at a numeric vector

B.235363_at a numeric vector

B.235369_at a numeric vector

B.235425_at a numeric vector

B.235542_at a numeric vector

B.235545_at a numeric vector

B.235547_at a numeric vector

B.235570_at a numeric vector

B.235572_at a numeric vector

B.235609_at a numeric vector

B.235662_at a numeric vector

B.235709_at a numeric vector

B.235771_at a numeric vector

B.235786_at a numeric vector

B.235789_at a numeric vector

B.235800_at a numeric vector

B.236050_at a numeric vector

B.236064_at a numeric vector

B.236312_at a numeric vector

B.236641_at a numeric vector

B.236787_at a numeric vector

B.236953_s_at a numeric vector

B.237086_at a numeric vector

B.237168_at a numeric vector

B.237301_at a numeric vector
B.237339_at a numeric vector
B.238001_at a numeric vector
B.238045_at a numeric vector
B.238075_at a numeric vector
B.238077_at a numeric vector
B.238116_at a numeric vector
B.238425_at a numeric vector
B.238447_at a numeric vector
B.238656_at a numeric vector
B.238756_at a numeric vector
B.238781_at a numeric vector
B.238898_at a numeric vector
B.239002_at a numeric vector
B.239349_at a numeric vector
B.239432_at a numeric vector
B.239758_at a numeric vector
B.239890_s_at a numeric vector
B.240099_at a numeric vector
B.240422_at a numeric vector
B.241310_at a numeric vector
B.241408_at a numeric vector
B.241577_at a numeric vector
B.241789_at a numeric vector
B.241937_s_at a numeric vector
B.242218_at a numeric vector
B.242255_at a numeric vector
B.242323_at a numeric vector
B.242560_at a numeric vector
B.242657_at a numeric vector
B.242890_at a numeric vector
B.243754_at a numeric vector
B.243806_at a numeric vector
B.244116_at a numeric vector
B.244375_at a numeric vector
B.244571_s_at a numeric vector
B.244677_at a numeric vector
B.244696_at a numeric vector
B.AFFX.HUMGAPDH.M33197_3_at a numeric vector
B.AFFX.HUMGAPDH.M33197_M_at a numeric vector
code a factor with levels case control

## Details

The factor code defines whether there was a mutation in the p53 sequence (code=case) or not (code=control).

## Source

Dr. Chris Holmes, c.holmes at stats dot. ox . ac .uk

## References

Miller et al (2005, PubMed ID:16141321)

## Examples

```
data(breastcancer)
## maybe str(breastcancer) ; plot(breastcancer) ...
```

---

# cad *Coronary artery disease data*

---

## Description

A cross classified table with observational data from a Danish heart clinic. The response variable is CAD.

## Usage

```
data(cad1)
```

## Format

A data frame with 236 observations on the following 14 variables.

`Sex` a factor with levels `Female` `Male`

`AngPec` a factor with levels `Atypical` `None` `Typical`

`AMI` a factor with levels `Definite` `NotCertain`

`QWave` a factor with levels `No` `Yes`

`QWavecode` a factor with levels `Nonusable` `Usable`

`STcode` a factor with levels `Nonusable` `Usable`

`STchange` a factor with levels `No` `Yes`

`SuffHeartF` a factor with levels `No` `Yes`

`Hypertrophi` a factor with levels `No` `Yes`

`Hyperchol` a factor with levels `No` `Yes`

`Smoker` a factor with levels `No` `Yes`

`Inherit` a factor with levels `No` `Yes`

`Heartfail` a factor with levels `No` `Yes`

`CAD` a factor with levels `No` `Yes`

## Details

cad1: Complete dataset, 236 cases. cad2: Incomplete dataset, 67 cases. Information on (some of) the variables Hyperchol, Smoker, Inherit is missing.

## References

Højsgaard, Søren and Thiesson, Bo (1995). BIFROST - Block recursive models Induced From Relevant knowledge, Observations and Statistical Techniques. Computational Statistics and Data Analysis, vol. 19, p. 155-175

Hansen, J. F. (1980). The clinical diagnoisis of ichaeme heart disease du to coronary artery disease. Danish Medical Bulletin

## Examples

```
data(cad1)
## maybe str(cad1) ; plot(cad1) ...
```

---

# carcass *Lean meat contents of 344 pig carcasses*

---

## Description

Measurement of lean meat percentage of 344 pig carcasses together with auxillary information collected at three Danish slaughter houses

## Usage

```
data(carcass)
data(carcassall)
```

## Format

carcassall: A data frame with 344 observations on the following 17 variables.

`weight` Weight of carcass

`lengthc` Length of carcass from back toe to head (when the carcass hangs in the back legs)

`lengthf` Length of carcass from back toe to front leg (that is, to the shoulder)

`lengthp` Length of carcass from back toe to the pelvic bone

`Fat02, Fat03, Fat11, Fat12, Fat13, Fat14, Fat16` Thickness of fat layer at different locations on the back of the carcass (FatXX refers to thickness at (or rather next to) rib no. XX. Notice that 02 is closest to the head

`Meat11, Meat12, Meat13` Thickness of meat layer at different locations on the back of the carcass, see description above

`LeanMeat` Lean meat percentage determined by dissection

`slhouse` Slaughter house; a factor with levels `a b c`

`sex` Sex of the pig; a factor with `a b c`. Notice that it is no an error to have three levels; the third level refers to castrates

carcass: Contains only the variables Fat11, Fat12, Fat13, Meat11, Meat12, Meat13, LeanMeat

## Source

Busk, H., Olsen, E. V., Brøndum, J. (1999) Determination of lean meat in pig carcasses with the Autofom classification system, Meat Science, 52, 307-314

---

`chestSim` *Simulated data from the Chest Clinic example*

---

## Description

Simulated data from the Chest Clinic example (also known as the Asia example) from Lauritzen and Spiegelhalter, 1988.

## Usage

```
data(chestSim500)
```

## Format

A data frame with 500 observations on the following 8 variables.

`asia` a factor with levels `yes no`

`tub` a factor with levels `yes no`

`smoke` a factor with levels `yes no`

`lung` a factor with levels `yes no`

`bronc` a factor with levels `yes no`

`either` a factor with levels `yes no`

`xray` a factor with levels `yes no`

`dysp` a factor with levels `yes no`

## References

Lauritzen and Spiegelhalter (1988) Local Computations with Probabilities on Graphical Structures and their Application to Expert Systems (with Discussion). J. Roy. Stat. Soc. 50, p. 157-224.

## Examples

```
data(chestSim500)
## maybe str(chestSim500) ; plot(chestSim500) ...
```

combnPrim *Generate All Combinations of n Elements Taken m at a Time*

## Description

Generate all combinations of the elements of x taken m at a time. If x is a positive integer, returns all combinations of the elements of seq(x) taken m at a time.

## Usage

```
combnPrim(x, m, simplify = TRUE)
```

## Arguments

| | |
|---|---|
| x | vector source for combinations, or integer n for x <- seq(n). |
| m | number of elements to choose. |
| simplify | logical indicating if the result should be simplified to a matrix; if FALSE, the function returns a list. |

## Value

A matrix or a list.

## Note

The combnPrim function is a simplified version of the combn function. However, combnPrim is implemented in C and is considerably faster than combn.

## Author(s)

P.T.Wallace and Søren Højsgaard

## See Also

combn

## Examples

```
x <- letters[1:20]
m <- 3

combn(x,m)
combnPrim(x,m)

combn(m,m)
combnPrim(m,m)
```

```
combn(x,m, simplify=FALSE)
combnPrim(x,m, simplify=FALSE)

system.time({ for (ii in 1:100) { combnPrim(x,m) }})
system.time({ for (ii in 1:100) { combn(x,m) }})

system.time({ for (ii in 1:100) { combnPrim(x,m, simplify=FALSE) }})
system.time({ for (ii in 1:100) { combn(x,m, simplify=FALSE) }})
```

---

`compareModels` *Generic function for model comparison*

---

## Description

`compareModels` is a generic functions which invoke particular methods which depend on the class of the first argument

## Usage

```
compareModels(object, object2, ...)
```

## Arguments

| | |
|---|---|
| `object, object2` | Model objects |
| `...` | Additional arguments |

## Value

The value returned depends on the class of the first argument.

## Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

---

`compile,propagate` *Compile and propagate functions*

---

## Description

`compile` and `propagate` are generic functions which invoke particular methods which depend on the class of the first argument

## Usage

```
compile(object, ...)
propagate(object, ...)
```

## Arguments

| | |
|---|---|
| object | An object to be compiled or propagated |
| ... | Additional arguments which depends on the class of the object |

## Value

The value returned depends on the class of the first argument.

## Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

## See Also

compile.grain, propagate.grain

---

# cov2pcor *Partial correlation (matrix)*

---

## Description

cov2pcor calculates the partial correlation matrix from an (empirical) covariance matrix while conc2pcor calculates the partial correlation matrix from a concentration matrix (inverse covariance matrix).

## Usage

```
cov2pcor(V)
conc2pcor(K)
```

## Arguments

| | |
|---|---|
| V | Covariance matrix |
| K | Concentration matrix |

## Value

A matrix with the same dimension as V.

## Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

## Examples

```
data(math)
S <- cov.wt(math)$cov
cov2pcor(S)
```

---

`dietox` *Growth curves of pigs in a 3x3 factorial experiment*

---

## Description

The `dietox` data frame has 861 rows and 7 columns.

## Usage

```
data(dietox)
```

## Format

This data frame contains the following columns: Weight, Feed, Time, Pig, Evit, Cu, Litter.

## Source

Lauridsen, C., Højsgaard, S.,Sørensen, M.T. C. (1999) Influence of Dietary Rapeseed Oli, Vitamin E, and Copper on Performance and Antioxidant and Oxidative Status of Pigs. J. Anim. Sci.77:906-916

## Examples

```
data(dietox)
```

---

`dumping` *Gastric Dumping*

---

## Description

A contingency table relating surgical operation, centre and severity of gastric dumping, a syndrome associated with gastric surgery.

## Usage

```
data(dumping)
```

## Format

A 3x4x4 table of counts cross-classified by Symptom (none/slight/moderate), Operation (Vd/Va/Vh/Gr) and Centre (1:4).

## Details

Gastric dumping syndrome is a condition where ingested foods bypass the stomach too rapidly and enter the small intestine largely undigested. It is an undesirable side-effect of gastric surgery. The table summarizes the results of a study comparing four different surgical operations on patients with duodenal ulcer, carried out in four centres, as described in Grizzle et al (1969). The four operations were: vagotomy and drainage, vagotomy and antrectomy (removal of 25% of gastric tissue), vagotomy and hemigastrectomy (removal of 50% of gastric tissue), and gastric restriction (removal of 75% of gastric tissue).

## Source

Grizzle JE, Starmer CF, Koch GG (1969) Analysis of categorical data by linear models. Biometrics 25(3):489-504.

## Examples

```
data(dumping)
plot(dumping)
```

---

# edgeList *Find edges in a graph and edges not in a graph.*

---

## Description

Returns the edges of a graph (or edges not in a graph) where the graph can be either a graphNEL object or an adjacency matrix.

## Usage

```
edgeList(object, matrix = FALSE)
## Default S3 method:
edgeList(object, matrix = FALSE)



nonEdgeList(object, matrix = FALSE)
## Default S3 method:
nonEdgeList(object, matrix = FALSE)



edgeListMAT(adjmat, matrix = FALSE)
nonEdgeListMAT(adjmat, matrix = FALSE)
```

## Arguments

| | |
|---|---|
| object | A graphNEL object or an adjacency matrix. |
| adjmat | An adjacency matrix. |
| matrix | If TRUE the result is a matrix; otherwise the result is a list. |

## Details

If object is a matrix, then edgeList() checks if object is symmetrical. If so it is assumed that the graph is undirected; otherwise the graph is assumed to be directed.

The workhorse is edgeListMAT.

## Value

A list or a matrix with edges.

## Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

## See Also

as.adjMAT mcs rip moralize jTree

## Examples

```
## A graph with edges
g  <- ug(~a:b+b:c+c:d)
gm <- as.adjMAT(g)

edgeList(g)
edgeList(gm)
edgeListMAT(gm)

edgeList(g, matrix=TRUE)
edgeList(gm, matrix=TRUE)
edgeListMAT(gm, matrix=TRUE)

nonEdgeList(g)
nonEdgeList(gm)
nonEdgeListMAT(gm)

## A graph without edges
g  <- ug(~a+b+c)
gm <- as.adjMAT(g)

edgeList(g)
edgeList(gm)
edgeListMAT(gm)

edgeList(g, matrix=TRUE)
```

```
edgeList(gm, matrix=TRUE)
edgeListMAT(gm, matrix=TRUE)

nonEdgeList(g)
nonEdgeList(gm)
nonEdgeListMAT(gm)
```

---

getCliques *Get cliques of an undirected graph*

---

## Description

Return a list of (maximal) cliques of an undirected graph.

## Usage

```
getCliques(object)
maxCliqueMAT(amat)
```

## Arguments

| | |
|---|---|
| object | An undirected graph represented either as a graphNEL object, a (dense) matrix, a (sparse) dgCMatrix |
| amat | An adjacency matrix |

## Details

I graph theory, a clique is often a complete subset of a graph. A maximal clique is a clique which can not be enlarged. In statistics (and that is the convention we follow here) a clique is usually understood to be a maximal clique.

Finding the cliques of a general graph is an NP complete problem. Finding the cliques of triangualted graph is linear in the number of cliques.

The workhorse is the maxCliqueMAT function which calls the maxClique function in the RBGL package.

## Value

A list

## Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

## See Also

ug dag mcs, mcsMAT rip, ripMAT, moralize, moralizeMAT

## Examples

```
## graphNEL
uG1 <- ug(~a:b+b:c+c:d+d:e+e:f+f:a)
getCliques(uG1)

## adjacency matrix
uG2 <- ug(~a:b+b:c+c:d+d:e+e:f+f:a, result="matrix")
getCliques(uG2)

## adjacency matrix (sparse)
uG3 <- ug(~a:b+b:c+c:d+d:e+e:f+f:a, result="Matrix")
getCliques(uG3)
```

---

`glist2adjMAT` *Creates adjacency matrix for a graph from a list of generators*

---

## Description

Creates adjacency matrix for a graph from a list of generators. The graph is assumed to be undirected.

## Usage

```
glist2adjMAT(glist, vn = unique(unlist(glist)), result="matrix")
vpaList2adjMAT(glist, vn = unique(unlist(glist)), result="matrix")
```

## Arguments

| | |
|---|---|
| `glist` | A list of generators where a generator is a vector of nodes. For `glist2adjMAT` a vector (v1,...,vn) means that there are undirected edges between all nodes. For `vpaList2adjMAT`, (v1,...,vn) means that there will be arrows from v2,...,vn to v1. |
| `vn` | The names of the vertices in the graphs. These will be the row and column names of the matrix |
| `result` | Either `"matrix"` or `"Matrix"` (for a sparse matrix representation) |

## Value

An adjacency matrix (or `NULL` if `glist` has length 0)

## Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

## See Also

ug dag

## Examples

```
glist <- list(1:3,2:4,4:5)
am1 <- glist2adjMAT(glist)
am2 <- vpaList2adjMAT(glist)
if (interactive()){
 plot(as(am1, "graphNEL"))
 plot(as(am2, "graphNEL"))
}
```

---

graph-coercion *Coercion of graphs*

---

## Description

Coercion of graphs from graphNEL format to adjacency matrix format and to other formats

## Usage

```
as.adjMAT(object, result="matrix")
graphNEL2adjMAT(object, result="matrix")
graphNEL2ftM(object)
graphNEL2matrix(object)
graphNEL2dgCMatrix(object)
```

## Arguments

| | |
|---|---|
| object | A graphNEL object |
| result | Either "matrix" of "Matrix" (for a sparse matrix representation). |

## Details

Notice: `graphNEL2adjMAT(g, result="matrix")` and `graphNEL2adjMAT(g, result="Matrix")` do the same as `as(g, "matrix")` and `as(g, "Matrix")` but considerably faster.

## Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

## See Also

edgeList getCliques nonEdgeList mcs rip moralize jTree

## Examples

```
uG <- ug(~me:ve,~me:al,~ve:al,~al:an,~al:st,~an:st)
getCliques(uG)

amat1 <- as.adjMAT(uG)
getCliques(amat1)

amat1 <- as.adjMAT(uG, result="Matrix")
getCliques(amat1)
```

---

# graph-operations *Simple operations on undirected and directed acyclic graphs.*

---

## Description

Make operations on undirected and directed acyclic graphs (which are represented as graphNEL objects.

## Usage

```
ancestors(set, object)
ancestralGraph(set, object)
ancestralSet(set, object)
children(set, object)
closure(set, object)
is.complete(object, set)
is.decomposition(set, set2, set3, object)
is.simplicial(set, object)
parents(set, object)
simplicialNodes(object)
```

## Arguments

| | |
|---|---|
| set, set2, set3 | Vectors of sets |
| object | A graphNEL object |

## Details

Notice: graphNEL2adjMAT(g, result="matrix") and graphNEL2adjMAT(g, result="Matrix") do the same as as(g, "matrix") and as(g, "Matrix") but considerably faster.

## Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

## See Also

edgeList nonEdgeList

mcs rip moralize jTree

## Examples

```
uG <- ug(~me:ve,~me:al,~ve:al,~al:an,~al:st,~an:st)
closure("me", uG)
getCliques(uG)

amat1 <- as.adjMAT(uG)
getCliques(amat1)

amat1 <- as.adjMAT(uG, result="Matrix")
getCliques(amat1)
```

---

gRbase *The package 'gRbase': summary information*

---

## Description

This package provides a basis for graphical modelling in R and in particular for other graphical modelling packages, most notably **gRim** and **gRain**. The package is a contribution to the gR-project described by Lauritzen (2002).

## Details

**gRbase** provides the following:

- Implementation of various graph algorithms, including maximum cardinality search, maximal prime subgraph decomposition, triangulation. See the vignette graphs.
- Implementation of various "high level" array operations, including multiplication/division, marginalization, slicing, permutation. See the vignette ArrayOps.
- Implementation of various "low level" array operations. See the vignette ArrayOpsPrim.
- A collection of datasets
- A general framework for setting up data and model structures and provide examples for fitting hierarchical log linear models for contingency tables and graphical Gaussian models for the multivariate normal distribution.

  Notice: This last part is not maintained / developed further.

## Authors

Søren Højsgaard, Aarhus University, DK-8830 Tjele, Denmark

Claus Dethlefsen, Center for Cardiovascular Research, Aalborg Hospital, Århus University Hospital, DK-9000 Aalborg, Denmark

Code for maximal prime subgraph decomposition has been provided by Clive Bowsher.

## Acknowledgements

Thanks to the other members of the gR initiative, in particular to David Edwards for providing functions for formula-manipulation.

## References

Lauritzen, S. L. (2002). gRaphical Models in R. *R News*, 3(2)39.

---

`gRbase-utilities` *Utility functions for gRbase*

---

## Description

Utility functions for gRbase package. Includes 'faster versions' of certain standard R functions.

## Details

`colwiseProd` multiplies a vector and a matrix columnwise (as opposed to rowwise which is achieved by `v*M`). Hence `colwiseProd` does the same as `t(v*t(M))` - but it does so faster for numeric values.

## Value

A vector or a logical.

## Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

## Examples

```
## colwiseProd
M <- matrix(1:16, nrow=4)
v <- 1:4

t(v*t(M))
colwiseProd(v,M)

system.time(for (ii in 1:100000)  t(v*t(M)))
system.time(for (ii in 1:100000)  colwiseProd(v,M))
```

## iplot *Function for plotting graphs using the 'igraph' package.*

### Description

Generic function for plotting graphs using the 'igraph' package and a plot method for graphNEL objects.

### Usage

```
iplot(x, ...)
## S3 method for class 'graphNEL'
iplot(x,...)
```

### Arguments

| | |
|---|---|
| x | A graph object to be plotted. |
| ... | Additional arguments |

### Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

### Examples

```
UG <- ug(~a:b+b:c:d)
iplot(UG)
```

## is.DAG *Check properties of graphs.*

### Description

Check if a graph is 1) a directed acyclic graph (DAG), 2) a directed graph (DG), 3) an undirected graph (UG), 4) a triangulated (chordal) undirected graph (TUG). This is done for graphs represented as 1) graphNEL (from the graph package), 2) an adjacency matrix, 3) a sparse adjacency matrix (a dgCMatrix from the Matrix package).

### Usage

```
is.DAG(object)
is.DG(object)
is.UG(object)
is.TUG(object)
is.adjMAT(x)
```

## Arguments

| | |
|---|---|
| object | A graph represented as 1) graphNEL (from the graph package), 2) an adjacency matrix, 3) a sparse adjacency matrix (a dgCMatrix from the Matrix package). |
| x | Any object. If the object is a quadratic matrix with 0's on the diagonal then it is an adjacency matrix. |

## Details

A non-zero value at entry (i,j) in an adjacency matrix A for a graph means that there is an edge from i to j. If also (j,i) is non-zero there is also an edge from j to i. In this case we may think of a bidirected edge between i and j or we may think of the edge as being undirected. We do not distinguish between undirected and bidirected edges in the gRbase package. On the other hand, graphNEL objects from the graph package makes such a distinction (the function `edgemode()` will tell if edges are "directed" or "undirected" in a graphNEL object).

The function `is.UG()` checks if the adjacency matrix is symmetric (If applied to a graphNEL, the adjacency matrix is created and checked for symmetry.)

The function `is.TUG()` checks if the graph is undirected and triangulated (also called chordal) by checking if the adjacency matrix is symmetric and the vertices can be given a perfect ordering using maximum cardinality seach.

The function `is.DG()` checks if a graph is directed, i.e., that there are no undirected edges. This is done by computing the elementwise product of A and the transpose of A; if there are no non–zero entries in this product then the graph is directed.

The function `is.DAG()` will return `TRUE` if all edges are directed and if there are no cycles in the graph. (This is checked by checking if the vertices in the graph can be given a topological ordering which is based on identifying an undirected edge with a bidrected edge).

There is a special case, namely if the graph has no edges at all (such that the adjacency matrix consists only of zeros). Such a graph is both undirected, triangulated, directed and directed acyclic.

## Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

## See Also

dag, ug

## Examples

```
## DAGs
dagNEL  <- dag(~a:b:c+c:d:e, result="NEL")
dagMAT  <- dag(~a:b:c+c:d:e, result="matrix")
dagMATS <- dag(~a:b:c+c:d:e, result="Matrix")

## Undirected graphs
ugNEL  <- ug(~a:b:c+c:d:e, result="NEL")
ugMAT  <- ug(~a:b:c+c:d:e, result="matrix")
ugMATS <- ug(~a:b:c+c:d:e, result="Matrix")
```

```
## Is it a DAG?
is.DAG(dagNEL)
is.DAG(dagMAT)
is.DAG(dagMATS)

is.DAG(ugNEL)
is.DAG(ugMAT)
is.DAG(ugMATS)

## Is it an undirected graph
is.UG(dagNEL)
is.UG(dagMAT)
is.UG(dagMATS)

is.UG(ugNEL)
is.UG(ugMAT)
is.UG(ugMATS)

## Is it a triangulated (i.e. chordal)  undirected graph
is.TUG(dagNEL)
is.TUG(dagMAT)
is.TUG(dagMATS)

is.TUG(ugNEL)
is.TUG(ugMAT)
is.TUG(ugMATS)

## Example where the graph is not triangulated
ug2NEL  <- ug(~a:b+b:c+c:d+d:a, result="NEL")
ug2MAT  <- ug(~a:b+b:c+c:d+d:a, result="matrix")
ug2MATS <- ug(~a:b+b:c+c:d+d:a, result="Matrix")

is.TUG(ug2NEL)
is.TUG(ug2MAT)
is.TUG(ug2MATS)


## Bidirected graphs
graph::edgemode(ugNEL)
graph::edgemode(ugNEL) <- "directed"
graph::edgemode(ugNEL)
is.DAG(ugNEL)
is.UG(ugNEL)
```

---

lizard *Lizard behaviour*

---

## Description

In a study of lizard behaviour, characteristics of 409 lizards were recorded, namely species (S), perch diameter (D) and perch height (H). The focus of interest is in how the propensities of the lizards to choose perch height and diameter are related, and whether and how these depend on species.

## Usage

```
data(lizard)
```

## Format

A 3–dimensional array with factors diam: "<=4" ">4" height: ">4.75" "<=4.75" species: "anoli" "dist"

## References

Schoener TW (1968) The anolis lizards of bimini: Resource partitioning in a complex fauna. Ecology 49:704-726

## Examples

```
data(lizard)

# Datasets lizardRAW and lizardDF are generated with the following code
#lizardAGG <- as.data.frame(lizard)
#f   <- lizardAGG$Freq
#idx <- unlist(mapply(function(i,n) rep(i,n),1:8,f))
#set.seed(0805)
#idx <- sample(idx)
#lizardRAW <- as.data.frame(lizardAGG[idx,1:3])
#rownames(lizardRAW) <- 1:NROW(lizardRAW)
```

---

# mathmark *Mathematics marks for students*

---

## Description

The mathmark data frame has 88 rows and 5 columns.

## Usage

```
data(mathmark)
```

## Format

This data frame contains the following columns: mechanics, vectors, algebra, analysis, statistics.

## Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

## References

David Edwards, An Introduction to Graphical Modelling, Second Edition, Springer Verlag, 2000

## Examples

```
data(mathmark)
```

---

mcs *Maximum cardinality search on undirected graph.*

---

## Description

Returns (if it exists) a perfect ordering of the vertices in an undirected graph.

## Usage

```
mcs(object, root = NULL, index = FALSE)
## Default S3 method:
mcs(object, root = NULL, index = FALSE)


mcsMAT(amat, vn = colnames(amat), root = NULL, index = FALSE)

mcsmarked(object, discrete = NULL, index = FALSE)
## Default S3 method:
mcsmarked(object, discrete = NULL, index = FALSE)


mcsmarkedMAT(amat, vn = colnames(amat), discrete = NULL, index = FALSE)
```

## Arguments

| | |
|---|---|
| object | An undirected graph represented either as a graphNEL object, an igraph, a (dense) matrix, a (sparse) dgCMatrix. |
| root | A vector of variables. The first variable in the perfect ordering will be the first variable on 'root'. The ordering of the variables given in 'root' will be followed as far as possible. |
| discrete | A vector indicating which of the nodes are discrete. See 'details' for more information. |
| index | If TRUE, then a permutation is returned |
| amat | Adjacency matrix |
| vn | Nodes in the graph given by adjacency matrix |

## Details

An undirected graph is decomposable iff there exists a perfect ordering of the vertices. The maximum cardinality search algorithm returns a perfect ordering of the vertices if it exists and hence this algorithm provides a check for decomposability. The `mcs()` functions finds such an ordering if it exists.

The notion of strong decomposability is used in connection with e.g. mixed interaction models where some vertices represent discrete variables and some represent continuous variables. Such graphs are said to be marked. The `mcsmarked()` function will return a perfect ordering iff the graph is strongly decomposable. As graphs do not know about whether vertices represent discrete or continuous variables, this information is supplied in the `discrete` argument.

## Value

A vector with a linear ordering (obtained by maximum cardinality search) of the variables or character(0) if such an ordering can not be created.

## Note

The workhorse is the `mcsMAT` function.

## Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

## See Also

`moralize` `jTree` `rip` `ug`, `dag`

## Examples

```
uG <- ug(~me+ve,~me+al,~ve+al,~al+an,~al+st,~an+st)
mcs(uG)
mcsMAT(as.adjMAT(uG))
## Same as
uG <- ug(~me+ve,~me+al,~ve+al,~al+an,~al+st,~an+st,result="matrix")
mcsMAT(uG)

## Marked graphs
uG1 <- ug(~a:b+b:c+c:d)
uG2 <- ug(~a:b+a:d+c:d)
## Not strongly decomposable:
mcsmarked(uG1, discrete=c("a","d"))
## Strongly decomposable:
mcsmarked(uG2, discrete=c("a","d"))
```

## mildew *Mildew fungus*

### Description

The data stem from a cross between two isolates of the barley powdery mildew fungus. For each offspring 6 binary characteristics, each corresponding to a single locus, were recorded. The object of the analysis is to determine the order of the loci along the chromosome.

### Usage

```
data(mildew)
```

### Format

The format is: table [1:2, 1:2, 1:2, 1:2, 1:2, 1:2] 0 0 0 0 3 0 1 0 0 1 ... - attr(*, "dimnames")=List of 6 ..$ la10: chr [1:2] "1" "2" ..$ locc: chr [1:2] "1" "2" ..$ mp58: chr [1:2] "1" "2" ..$ c365: chr [1:2] "1" "2" ..$ p53a: chr [1:2] "1" "2" ..$ a367: chr [1:2] "1" "2"

### References

Christiansen, S.K., Giese, H (1991) Genetic analysis of obligate barley powdery mildew fungus based on RFLP and virulence loci. Theor. Appl. Genet. 79:705-712

### Examples

```
data(mildew)
## maybe str(mildew) ; plot(mildew) ...
```

## milkcomp *Milk composition data*

### Description

Data from an experiment on composition of sow milk. Milk composition is measured on four occasions during lactation on a number of sows. The treatments are different types of fat added to the sows feed.

### Usage

```
data(milkcomp)
```

## Format

A data frame with 214 observations on the following 7 variables.

`sow` a numeric vector

`lactime` a numeric vector

`treat` a factor with levels `a b c d e f g`

`fat` a numeric vector

`protein` a numeric vector

`dm` (dry matter) a numeric vector

`lactose` a numeric vector

## Details

`a` is the control, i.e. no fat has been added.

`fat` + `protein` + `lactose` almost add up to `dm` (dry matter)

## References

Charlotte Lauridsen and Viggo Danielsen (2004): Lactational dietary fat levels and sources influence milk composition and performance of sows and their progeny Livestock Production Science 91 (2004) 95-105

## Examples

```
data(milkcomp)
## maybe str(milk) ; plot(milk) ...
```

---

`minimalTriang` *Minimal triangulation of an undirected graph*

---

## Description

An undirected graph uG is triangulated (or chordal) if it has no cycles of length >= 4 without a chord which is equivalent to that the vertices can be given a perfect ordering. Any undirected graph can be triangulated by adding edges to the graph, so called fill-ins which gives the graph TuG. A triangulation TuG is minimal if no fill-ins can be removed without breaking the property that TuG is triangulated.

## Usage

```
minimalTriang(object, tobject = triangulate(object), result=NULL, details = 0)
minimalTriangMAT(amat, tamat = triangulateMAT(amat), details = 0)
```

## Arguments

| | |
|---|---|
| object | An undirected graph represented either as a graphNEL object, a (dense) matrix, a (sparse) dgCMatrix. |
| tobject | Any triangulation of object; must be of the same representation. |
| result | The type (representation) of the result. Possible values are "graphNEL", "matrix", "dgCMatrix". Default is the same as the type of object. |
| amat | The undirected graph which is to be triangulated; a symmetric adjacency matrix |
| tamat | Any triangulation of object; a symmetric adjacency matrix |
| details | The amount of details to be printed. |

## Details

For a given triangulation tobject it may be so that some of the fill-ins are superflous in the sense that they can be removed from tobject without breaking the property that tobject is triangulated. The graph obtained by doing so is a minimal triangulation.

Notice: A related concept is the minimum triangulation, which is the the graph with the smallest number of fill-ins. The minimum triangulation is unique. Finding the minimum triangulation is NP-hard.

## Value

minimalTriang() returns a graphNEL object while minimalTriangMAT() returns an adjacency matrix.

## Author(s)

Clive Bowsher <C.Bowsher@statslab.cam.ac.uk> with modifications by Søren Højsgaard, <sorenh@math.aau.dk>

## References

Kristian G. Olesen and Anders L. Madsen (2002): Maximal Prime Subgraph Decomposition of Bayesian Networks. IEEE TRANSACTIONS ON SYSTEMS, MAN AND CYBERNETICS, PART B: CYBERNETICS, VOL. 32, NO. 1, FEBRUARY 2002

## See Also

mpd,rip, triangulate

## Examples

```
## A graphNEL object
g1 <- ug(~a:b+b:c+c:d+d:e+e:f+a:f+b:e)
x <- minimalTriang(g1)

## g2 is a triangulation of g1 but it is not minimal
g2 <- ug(~a:b:e:f+b:c:d:e)
x<-minimalTriang(g1, tobject=g2)
```

```
## An adjacency matrix
g1m <- ug(~a:b+b:c+c:d+d:e+e:f+a:f+b:e, result="matrix")
x<-minimalTriangMAT(g1m)
```

---

`moralize` *Moralize a directed acyclic graph*

---

## Description

Moralize a directed acyclic graph which means marrying parents and dropping directions

## Usage

```
moralize(object, ...)
## Default S3 method:
moralize(object, result=NULL, ...)
```

## Arguments

| | |
|---|---|
| `object` | A directed acyclic graph represented either as a `graphNEL` object, an `igraph`, a (dense) `matrix`, a (sparse) `dgCMatrix`. |
| `result` | The representation of the moralized graph. When NULL the representation will be the same as the input object. |
| `...` | Additional arguments, currently not used |

## Value

A moralized graph represented either as a 'graphNEL', a 'matrix' or a sparse 'dgCMatrix'.

## Note

The workhorse is the `moralizeMAT` function.

## Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

## See Also

`mcs` `jTree` `rip` `ug`, `dag`

## Examples

```
daG <- dag(~me+ve,~me+al,~ve+al,~al+an,~al+st,~an+st)
moralize(daG)

daG <- dag(~me+ve,~me+al,~ve+al,~al+an,~al+st,~an+st, result="matrix")
moralizeMAT(daG)
```

---

mpd *Maximal prime subgraph decomposition*

---

## Description

Finding a junction tree representation of the MPD (maximal prime subgraph decomposition) of an undirected graph

## Usage

```
mpd(object, tobject = minimalTriang(object), details = 0)
mpdMAT(amat, tamat = minimalTriangMAT(amat), details = 0)
```

## Arguments

| | |
|---|---|
| object | An undirected graph; a graphNEL object |
| tobject | Any minimal triangulation of object; a graphNEL object |
| amat | An undirected graph; a symmetric adjacency matrix |
| tamat | Any minimal triangulation of object; a symmetric adjacency matrix |
| details | The amount of details to be printed. |

## Details

The maximal prime subgraph decomposition of a graph is the smallest subgraphs into which the graph can be decomposed.

## Value

A list with components "nodes", "cliques", "separators", "parents", "children", "nLevels". The component "cliques" defines the subgraphs.

## Author(s)

Clive Bowsher <C.Bowsher@statslab.cam.ac.uk> with modifications by Søren Højsgaard, <sorenh@math.aau.dk>

## References

Kristian G. Olesen and Anders L. Madsen (2002): Maximal Prime Subgraph Decomposition of Bayesian Networks. IEEE TRANSACTIONS ON SYSTEMS, MAN AND CYBERNETICS, PART B: CYBERNETICS, VOL. 32, NO. 1, FEBRUARY 2002

## See Also

mcs, mcsMAT, minimalTriang, minimalTriangMAT, rip, ripMAT, triangulate, triangulateMAT

## Examples

```
## Maximal prime subgraph decomposition - a graphNEL object
g1 <- ug(~a:b+b:c+c:d+d:e+e:f+a:f+b:e)
if (interactive()) plot(g1)
x <- mpd(g1)

## Maximal prime subgraph decomposition - an adjacency matrix
g1m <- ug(~a:b+b:c+c:d+d:e+e:f+a:f+b:e, result="matrix")
if (interactive()) plot(as(g1m, "graphNEL"))
x <- mpdMAT(g1m)
```

---

# Nutrimouse *The Nutrimouse Dataset*

---

## Description

The data come from a study of the effects of five dietary regimens with different fatty acid compositions on liver lipids and hepatic gene expression in 40 mice.

## Usage

```
data(Nutrimouse)
```

## Format

A data frame with 40 observations on the following 143 variables.

`genotype` a factor with levels `wt` `ppar`

`diet` a factor with levels `coc` `fish` `lin` `ref` `sun`

`X36b4` a numeric vector

`ACAT1` a numeric vector

`ACAT2` a numeric vector

`ACBP` a numeric vector

`ACC1` a numeric vector

`ACC2` a numeric vector

`ACOTH` a numeric vector

`ADISP` a numeric vector

`ADSS1` a numeric vector

`ALDH3` a numeric vector

AM2R a numeric vector

AOX a numeric vector

BACT a numeric vector

BIEN a numeric vector

BSEP a numeric vector

Bcl.3 a numeric vector

C16SR a numeric vector

CACP a numeric vector

CAR1 a numeric vector

CBS a numeric vector

CIDEA a numeric vector

COX1 a numeric vector

COX2 a numeric vector

CPT2 a numeric vector

CYP24 a numeric vector

CYP26 a numeric vector

CYP27a1 a numeric vector

CYP27b1 a numeric vector

CYP2b10 a numeric vector

CYP2b13 a numeric vector

CYP2c29 a numeric vector

CYP3A11 a numeric vector

CYP4A10 a numeric vector

CYP4A14 a numeric vector

CYP7a a numeric vector

CYP8b1 a numeric vector

FAS a numeric vector

FAT a numeric vector

FDFT a numeric vector

FXR a numeric vector

G6PDH a numeric vector

G6Pase a numeric vector

GK a numeric vector

GS a numeric vector

GSTa a numeric vector

GSTmu a numeric vector

GSTpi2 a numeric vector

HMGCoAred a numeric vector

HPNCL a numeric vector

IL.2 a numeric vector

L.FABP a numeric vector

LCE a numeric vector

LDLr a numeric vector

LPK a numeric vector

LPL a numeric vector

LXRa a numeric vector

LXRb a numeric vector

Lpin a numeric vector

Lpin1 a numeric vector

Lpin2 a numeric vector

Lpin3 a numeric vector

M.CPT1 a numeric vector

MCAD a numeric vector

MDR1 a numeric vector

MDR2 a numeric vector

MRP6 a numeric vector

MS a numeric vector

MTHFR a numeric vector

NGFiB a numeric vector

NURR1 a numeric vector

Ntcp a numeric vector

OCTN2 a numeric vector

PAL a numeric vector

PDK4 a numeric vector

PECI a numeric vector

PLTP a numeric vector

PMDCI a numeric vector

PON a numeric vector

PPARa a numeric vector

PPARd a numeric vector

PPARg a numeric vector

PXR a numeric vector

Pex11a a numeric vector

RARa a numeric vector

RARb2 a numeric vector

RXRa a numeric vector

RXRb2 a numeric vector

RXRg1 a numeric vector

S14 a numeric vector

SHP1 a numeric vector

SIAT4c a numeric vector

SPI1.1 a numeric vector

SR.BI a numeric vector

THB a numeric vector

THIOL a numeric vector

TRa a numeric vector

TRb a numeric vector

Tpalpha a numeric vector

Tpbeta a numeric vector

UCP2 a numeric vector

UCP3 a numeric vector

VDR a numeric vector

VLDLr a numeric vector

Waf1 a numeric vector

ap2 a numeric vector

apoA.I a numeric vector

apoB a numeric vector

apoC3 a numeric vector

apoE a numeric vector

c.fos a numeric vector

cHMGCoAS a numeric vector

cMOAT a numeric vector

eif2g a numeric vector

hABC1 a numeric vector

i.BABP a numeric vector

i.BAT a numeric vector

i.FABP a numeric vector

i.NOS a numeric vector

mABC1 a numeric vector

mHMGCoAS a numeric vector

C14.0 a numeric vector

`C16.0` a numeric vector

`C18.0` a numeric vector

`C16.1n.9` a numeric vector

`C16.1n.7` a numeric vector

`C18.1n.9` a numeric vector

`C18.1n.7` a numeric vector

`C20.1n.9` a numeric vector

`C20.3n.9` a numeric vector

`C18.2n.6` a numeric vector

`C18.3n.6` a numeric vector

`C20.2n.6` a numeric vector

`C20.3n.6` a numeric vector

`C20.4n.6` a numeric vector

`C22.4n.6` a numeric vector

`C22.5n.6` a numeric vector

`C18.3n.3` a numeric vector

`C20.3n.3` a numeric vector

`C20.5n.3` a numeric vector

`C22.5n.3` a numeric vector

`C22.6n.3` a numeric vector

## Details

The data come from a study of the effects of five dietary regimens with different fatty acid compositions on liver lipids and hepatic gene expression in wild-type and PPAR-alpha-deficient mice (Martin et al., 2007). There were 5 replicates per genotype and diet combination.

There are two design variables: (i) genotype, a factor with two levels: wild-type (wt) and PPAR-alpha-deficient (ppar), and (ii) diet, a factor with five levels. The oils used for experimental diet preparation were: corn and colza oils (50/50) for a reference diet (ref); hydrogenated coconut oil for a saturated fatty acid diet (coc); sunflower oil for an Omega6 fatty acid-rich diet (sun); linseed oil for an Omega3-rich diet (lin); and corn/colza/enriched (43/43/14) fish oils (fish).

There are 141 response variables: (i) the log-expression levels of 120 genes measured in liver cells, and (ii) the concentrations (in percentages) of 21 hepatic fatty acids measured by gas chromatography.

## Source

The data were provided by Pascal Martin from the Toxicology and Pharmacology Laboratory, National Institute for Agronomic Research, French.

## References

Martin, P. G. P., Guillou, H., Lasserre, F., Déjean, S., Lan, A., Pascussi, J.-M., San Cristobal, M., Legrand, P., Besse, P. and Pineau, T. (2007). Novel aspects of PPARa-mediated regulation of lipid and xenobiotic metabolism revealed through a multrigenomic study. Hepatology 54, 767-777.

## Examples

```
data(Nutrimouse)
```

---

`parray` *Representation of and operations on multidimensional tables*

---

## Description

General representation of multidimensional tables (by `parray` objects).

## Usage

```
parray(varNames, levels, values = 1, normalize = "none", smooth = 0)
as.parray(values, normalize="none", smooth=0)
data2parray(data, varNames=NULL, normalize="none", smooth=0)
```

## Arguments

| | |
|---|---|
| `varNames` | Names of variables defining table; can be a right hand sided formula. |
| `levels` | Either 1) a vector with number of levels of the factors in varNames or 2) a list with specification of the levels of the factors in varNames. See 'examples' below. |
| `values` | Values to go into the parray |
| `normalize` | Either "none", "first" or "all". Should result be normalized, see 'Details' below. |
| `smooth` | Should values be smoothed, see 'Details' below. |
| `data` | A dataframe, a table, an xtabs (a cross classified contingency table) a matrix (with dimnames) or a vector (with dimnames). |

## Details

A `parray` object represents a table defined by a set of variables and their levels, together with the values of the table. E.g. f(a,b,c) can be a table with a,b,c representing levels of binary variable

If `normalize="first"` then for each configuration of all other variables than the first, the probabilities are normalized to sum to one. Thus f(a,b,c) becomes a conditional probability table of the form p(a|b,c). If `normalize="all"` then the sum over all entries of f(a,b,c) is one.

If `smooth` is positive then `smooth` is added to `values` before normalization takes place.

`as.parray` can be used for coercing an array or an xtabs to a parray object.

## Value

An object of class parray.

## Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

## See Also

tableOp, tableMargin

## Examples

```
t1 <- parray(c("gender","answer"), list(c('male','female'),c('yes','no')), values=1:4)
t1 <- parray(~gender:answer, list(c('male','female'),c('yes','no')), values=1:4)
t1 <- parray(~gender:answer, c(2,2), values=1:4)

t2 <- parray(c("answer","category"), list(c('yes','no'),c(1,2)), values=1:4+10)
t3 <- parray(c("category","foo"), c(2,2), values=1:4+100)

varNames(t1)
nLevels(t1)
valueLabels(t1)


## Create 1-dimensional vector with dim and dimnames
x1 <- 1:5
as.parray(x1)
x2 <- parray("x", levels=length(x1), values=x1)
dim(x2)
dimnames(x2)

## Matrix
x1 <- matrix(1:6, nrow=2)
as.parray(x1)
parray(~a:b, levels=dim(x1), values=x1)

## Extract parrays from data
## 1) a dataframe
data(cad1)
data2parray(cad1, ~Sex:AngPec:AMI)
data2parray(cad1, c("Sex","AngPec","AMI"))
data2parray(cad1, c(1,2,3))
## 2) a table
data2parray(UCBAdmissions,c(1,2), normalize="first")
```

querygraph *Query a graph*

## Description

queryg is a general function for querying a graph object, specifically graphs as created with newug and newdag.

## Usage

```
querygraph(object, op, set = NULL, set2 = NULL, set3 = NULL)
```

## Arguments

| | |
|---|---|
| object | A graph object; i.e. either an unidirected graph (ugsh) or a directed acyclic graph (dagsh) |
| op | The query operation, see 'details' below. |
| set,set2,set3 | Possible arguments to a graph query of type type |

## Details

The op can be:

- adj: Nodes adjacant to set
- an: Ancestors of set
- ancestralGraph: Ancestral graph induced by set
- ancestralSet: Ancestral set of set
- cl: Closure of set
- ch: Children of set
- maxClique: The cliques
- connectedComp The connected components
- edges: Edges of graph
- ne: Neighbours of set
- nodes: Nodes of graph
- is.complete:
- edgeList
- vpar
- is.simplicial:
- is.triangulated:
- pa: Parents of set
- separates: Is set and set2 separated by set3
- simplicialNodes: The simplicial nodes of graph
- subgraph: Subgraph induced by set

## Value

Depending on the type, the output will be either a new graph or a vector or a list.

## Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

## See Also

ug, dag

## Examples

```
ug0 <- ug(~a:b, ~b:c:d, ~e)

querygraph(ug0, "nodes")
querygraph(ug0, "edges")

querygraph(ug0, "subgraph", c("b","c","d","e"))

querygraph(ug0, "adj", "c")
querygraph(ug0, "closure", "c")
querygraph(ug0, "is.simplicial", "b")
querygraph(ug0, "simplicialNodes")

querygraph(ug0, "is.complete")
querygraph(ug0, "is.complete", c("b","c","d"))
querygraph(ug0, "maxClique")

querygraph(ug0, "is.triangulated")
querygraph(ug0, "is.decomposition", "a","d",c("b","c"))
```

---

# random_dag *Random directed acyclic graph*

---

## Description

Generate a random directed acyclic graph (DAG)

## Usage

```
random_dag(V, maxpar = 3, wgt = 0.1)
```

### Arguments

| | |
|---|---|
| V | The set of vertices. |
| maxpar | The maximum number of parents each node can have |
| wgt | A parameter controlling how likely it is for a node to have a certain number of parents; see 'Details' |

### Details

If the maximum number of parents for a node is, say 3 and wgt=0.1, then the probability of the node ending up with 0,1,2,3 parents is proportional to 0.1^0, 0.1^1, 0.1^2, 0.1^3.

### Value

A graphNEL object.

### Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

### Examples

```
dg   <- random_dag(1:1000, maxpar=5, wgt=.9)
table(sapply(vpar(dg),length))

dg   <- random_dag(1:1000, maxpar=5, wgt=.5)
table(sapply(vpar(dg),length))

dg   <- random_dag(1:1000, maxpar=5, wgt=.1)
table(sapply(vpar(dg),length))
```

---

## rats *Weightloss of rats*

---

### Description

An artificial dataset. 24 rats (12 female, 12 male) have been randomized to use one of three drugs (products for loosing weight). The weightloss for each rat is noted after one and two weeks.

### Usage

```
data(rats)
```

### Format

A dataframe with 4 variables. Sex: "M" (male), "F" (female). Drug: "D1", "D2", "D3" (three types). W1 weightloss, week one. W2 weightloss, week 2.

## References

Morrison, D.F. (1976). Multivariate Statistical Methods. McGraw-Hill, USA.

Edwards, D. (1995). Introduction to Graphical Modelling, Springer-Verlag. New York.

---

`reinis` *Risk factors for coronary heart disease.*

---

## Description

Data collected at the beginning of a 15 year follow-up study of probable risk factors for coronary thrombosis. Data are from all men employed in a car factory.

## Usage

```
data(reinis)
```

## Format

A table with 6 discrete variables. A: smoking, B: strenous mental work, D: strenuous physical work, E: systolic blood pressure, F: ratio of lipoproteins, G: Family anamnesis of coronary heart disease.

## References

Edwards and Havranek (1985): A fast procedure for model search in multidimensional contingency tables. Biometrika, 72: 339-351.

Reinis et al (1981): Prognostic significance of the risk profile in the prevention of coronary heart disease. Bratis. lek. Listy. 76: 137-150.

---

`rip` *Create RIP ordering of the cliques of an undirected graph; create junction tree.*

---

## Description

A RIP (running intersection property) ordering of the cliques is also called a perfect ordering. If the graph is not chordal, then no such ordering exists.

## Usage

```
rip(object, root = NULL, nLevels = NULL)
## Default S3 method:
rip(object, root = NULL, nLevels = NULL)



ripMAT(amat, root = NULL, nLevels = rep(2, ncol(amat)))

jTree(object, ...)
## Default S3 method:
jTree(object, nLevels=NULL, ...)



jTreeMAT(amat, nLevels=rep(2,ncol(amat)), ...)
```

## Arguments

| | |
|---|---|
| object | An undirected graph represented either as a graphNEL object, an igraph, a (dense) matrix, a (sparse) dgCMatrix. |
| root | A vector of variables. The first variable in the perfect ordering will be the first variable on 'root'. The ordering of the variables given in 'root' will be followed as far as possible. |
| nLevels | Typically, the number of levels of the variables (nodes) when these are discrete. Used in determining the triangulation using a "minimum clique weight heuristic". See section 'details'. |
| amat | Adjacency matrix |
| ... | Additional arguments; currently not used |

## Details

The RIP ordering of the cliques of a decomposable (i.e. chordal) graph is obtained by first ordering the variables linearly with maximum cardinality search (by mcs). The root argument is transfered to mcs as a way of controlling which clique will be the first in the RIP ordering.

The jTree() (and jTree()) (for "junction tree") is just a wrapper for a call of triangulate() followed by a call of rip().

## Value

rip returns a list (an object of class ripOrder. A print method exists for such objects.)

## Note

The workhorse is the ripMAT() function.

The nLevels argument to the rip functions has no meaning.

### Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

### See Also

mcs triangulate moralize ug, dag

### Examples

```
## graphNEL
uG <- ug(~me:ve + me:al + ve:al + al:an + al:st + an:st)
mcs(uG)
rip(uG)
jTree(uG)

## Adjacency matrix
uG <- ug(~me:ve:al + al:an:st, result="matrix")
mcs(uG)
rip(uG)
jTree(uG)

## Sparse adjacency matrix
uG <- ug(c("me","ve","al"),c("al","an","st"), result="Matrix")
mcs(uG)
rip(uG)
jTree(uG)

## Non--decomposable graph
uG <- ug(~1:2+2:3+3:4+4:5+5:1)
mcs(uG)
rip(uG)
jTree(uG)
```

---

## Setoperations *Set operations*

---

### Description

Miscellaneous set operations.

### Usage

```
is.subsetof(x, set)
is.insetlist(x, setlist, index=FALSE)
removeRedundant(setlist, maximal = TRUE, index = FALSE)
maximalSets(setlist,  index = FALSE)
minimalSets(setlist,  index = FALSE)
```

### Arguments

| | |
|---|---|
| x, set | Vectors representing sets |
| setlist | List of vectors (representing a set of subsets) |
| maximal | Logical; see section 'Details' for a description. |
| index | Logical; should indices (in setlist) be returned or a set of subsets. |

### Details

'setlist' is a list of vectors representing a set of subsets; i.e. V1,...VQ where Vk is a subset of some base set V.

is.insetlist: Checks if the set x is in one of the Vk's.

removeRedundant: Returns those Vk which are not contained in other subsets; i.e. gives the maximal sets. If maximal is FALSE then returns the minimal sets; i.e. Vk is returned if Vk is contained in one of the other sets Vl and there are no set Vn contained in Vk.

Notice that the comparisons are made by turning the elements into characters and then comparing these. Hence 1 is identical to "1".

### Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

### Examples

```
is.subsetof(c(1,2),c(1,2,3))
is.subsetof(c(1,2,3), c(1,2))

l <- list(c(1,2),c(1,2,3),c(2,4),c(5,6), 5)

#subsetofList(c(1,2), l)
#subsetofList(c(1,2,3,4), l)

removeRedundant(l)
removeRedundant(l, maximal=FALSE)

is.insetlist (c(2,4), l)
is.insetlist (c(2,8), l)
```

---

## simulateArray *Simulate data from array*

---

### Description

Simulate data (slice of) an array.

## Usage

```
simulateArray(x, nsim = 1, margin, value.margin)
```

## Arguments

| | |
|---|---|
| x | An array |
| nsim | Number of cases to simulate |
| margin, value.margin | Specification of slice of array to simulate from |

## Value

A matrix

## Note

The current implementation is fragile in the sense that it is not checked that the input argument x is an array.

## Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

## Examples

```
## 2x2 array
x <- parray(c("a","b"), levels=c(2,2), values=1:4)

## Simulate from entire array
s <-simulateArray(x,1000)
xtabs(~., as.data.frame(s))

## Simulate from slice defined by that dimension 1 is fixed at level 2
s <-simulateArray(x,6000,1,2)
xtabs(~., as.data.frame(s))

## 2x2x2 array
x <- parray(c("a","b","c"), levels=c(2,2,2), values=1:8)

## Simulate from entire array
s <-simulateArray(x,36000)
xtabs(~., as.data.frame(s))

## Simulate from slice defined by that dimension 3 is fixed at level 1
s <-simulateArray(x,10000,3,1)
xtabs(~., as.data.frame(s))
```

table-operations *Compute table margin or table slice*

## Description

For a contingency table in array form, compute the sum of table entries for a given index (i.e. a marginal table) or find the slice of the table defined by specific margins being at a specific level.

## Usage

```
tableOp(t1, t2, op = "*")
tableMult(t1, t2)
tableDiv(t1, t2)
tableMargin(x, margin, keep.class=FALSE)
tableSlice(x, margin, level, impose)
tablePerm(a, perm, resize = TRUE, keep.class=FALSE)
```

## Arguments

| | |
|---|---|
| x, t1, t2, a | An array |
| margin | An index, either numerical or character |
| keep.class | If TRUE the result will be forced to have the same class as the input; otherwise the result will be an array. |
| level | A value, either numerical or character |
| impose | Possible value used to fill up a slice to give it full dimension |
| op | Either "*" or "/" |
| perm | The subscript permutation vector, which must be a permutation of the integers 1:n, where n is the number of dimensions of a OR a permutation of the dimension names of a. The default is to reverse the order of the dimensions. A permutation of the dimensions of a. |
| resize | A flag indicating whether the vector should be resized as well as having its elements reordered. |

## Details

tableMargin: tableMargin is analogous to margin.table except that margin can be given both as array indices or as variable names

tableSlice: If the table x has dimensions Z,U,V where V has levels 1 and 2 then tableSlice can extract the slice of x (in this case a 2-way table) defined by e.g. U=2. Setting impose=1000 implies that a 3-way table is returned with the U=2 slice in the right place and the U=1-slice consisting of 1000 in each cell.

tableOp: If t1 has dimnames A and B and t2 has dimnames B and C then tableOp(t1,t2) will return a table (an array) with dimnames A, B and C containing the product.

`tableMult(t1,t2)` is a wrapper for `tableOp(t1,t2, op="*")` and `tableMult(t1,t2)` is a wrapper for `tableDiv(t1,t2, op="/")`

`tablePerm`: A wrapper for aperm, but tablePerm accepts dimnames in addition to indices.

See examples below.

## Value

An array.

## Author(s)

Søren Højsgaard

## See Also

margin.table

## Examples

```
data(HairEyeColor)

tableMargin(HairEyeColor, "Hair")
tableMargin(HairEyeColor, 1)
tableMargin(HairEyeColor, c("Hair","Eye"))
tableMargin(HairEyeColor, c(1,2))


tableSlice(HairEyeColor, "Sex","Male")
tableSlice(HairEyeColor, 3,1)
tableSlice(HairEyeColor, "Sex","Male", impose=1000)
tableSlice(HairEyeColor, 3,1, impose=1000)

t1 <- array(1:4, dim=c(2,2), dimnames=list(gender=c('male','female'),answer=c('yes','no')))
t2 <- array(1:4+10, dim=c(2,2), dimnames=list(answer=c('yes','no'),category=c(1,2)))

tableOp(t1,t2, "*")
tableOp(t1,t2, "/")

data(reinis)

t1 <- tableMargin(reinis, c(6,5,2,1))
t2 <- tableMargin(reinis, c(6,5,3,4))

tt1 <- tableOp(t1,t2)

t1 <- tableMargin(reinis, c(6,5,2,4,1))
t2 <- tableMargin(reinis, c(6,5,4))

tt1 <- tableOp2(t1,t2)
```

topoSort *Topological sort of vertices in directed acyclic graph*

## Description

A topological ordering of a directed graph is a linear ordering of its vertices such that, for every edge (u->v), u comes before v in the ordering. A topological ordering is possible if and only if the graph has no directed cycles, that is, if it is a directed acyclic graph (DAG). Any DAG has at least one topological ordering. Can hence be used for checking if a graph is a DAG.

## Usage

```
topoSort(object, index = FALSE)
```

## Arguments

| | |
|---|---|
| object | An graph represented either as a graphNEL object, an igraph, a (dense) matrix, a (sparse) dgCMatrix. |
| index | If FALSE, an ordering is returned if it exists and character(0) otherwise. If TRUE, the index of the variables in an adjacency matrix is returned and -1 otherwise. |

## Value

If FALSE, an ordering is returned if it exists and character(0) otherwise. If TRUE, the index of the variables in an adjacency matrix is returned and -1 otherwise.

## Note

The workhorse is the topoSortMAT function which takes an adjacency matrix as input

## Warning

Do not use index=TRUE when the input is a graphNEL object; the result is unpredictable.

## Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

## See Also

dag, ug

## Examples

```
dagMAT  <- dag(~a:b:c+c:d:e, result="matrix")
dagMATS <- as(dagMAT, "dgCMatrix")
dagNEL  <- as(dagMAT, "graphNEL")

topoSort(dagMAT)
topoSort(dagMATS)
topoSort(dagNEL)
```

---

# triangulate *Triangulation of an undirected graph*

---

## Description

This function will triangulate an undirected graph by adding fill-ins.

## Usage

```
triangulate(object, ...)
## Default S3 method:
triangulate(object, nLevels = NULL, result=NULL, ...)
triangulateMAT(amat, nLevels=rep(2, ncol(amat)), ...)
```

## Arguments

| | |
|---|---|
| object | An undirected graph represented either as a graphNEL object, an igraph, a (dense) matrix, a (sparse) dgCMatrix. |
| nLevels | The number of levels of the variables (nodes) when these are discrete. Used in determining the triangulation using a "minimum clique weight heuristic". See section 'details'. |
| result | The type (representation) of the result. Possible values are "graphNEL", "igraph", "matrix", "dgCMatrix". Default is the same as the type of object. |
| ... | Additional arguments, currently not used. |
| amat | Adjacency matrix; a (dense) matrix, or a (sparse) dgCMatrix. |

## Details

The workhorse is the triangulateMAT function.

The triangulation is made so as the total state space is kept low by applying a minimum clique weight heuristic: When a fill-in is necessary, the algorithm will search for an edge to add such that the complete set to be formed will have as small a state-space as possible. It is in this connection that the nLevels values are used.

Default (when nLevels=NULL) is to take nLevels=2 for all nodes. If nLevels is the same for all nodes then the heuristic aims at keeping the clique sizes small.

## Value

A triangulated graph represented either as a graphNEL, a (dense) matrix or a (sparse) dgCMatrix.

## Note

Care should be taken when specifying nLevels for other representations than adjacency matrices: Since the triangulateMAT function is the workhorse, any other representation is transformed to an adjacency matrix and the order of values in nLevels most come in the order of the nodes in the adjacency matrix representation.

Currently there is no check for that the graph is undirected.

## Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

## See Also

ug dag mcs, mcsMAT rip, ripMAT, moralize, moralizeMAT

## Examples

```
## graphNEL
uG1 <- ug(~a:b+b:c+c:d+d:e+e:f+f:a)
tuG1 <- triangulate(uG1)

## adjacency matrix
uG2 <- ug(~a:b+b:c+c:d+d:e+e:f+f:a, result="matrix")
tuG2 <- triangulate(uG2)

## adjacency matrix (sparse)
uG2 <- ug(~a:b+b:c+c:d+d:e+e:f+f:a, result="Matrix")
tuG2 <- triangulate(uG2)
```

---

# ug *Create undirected and directed graphs*

---

## Description

These functions are wrappers for creation of graphs as implemented by graphNEL objects in the graph package.

## Usage

```
ug(..., result="NEL")
dag(..., result="NEL", forceCheck=FALSE)
ugList(x, result="NEL")
dagList(x, result="NEL", forceCheck=FALSE)
```

## Arguments

| | |
|---|---|
| ... | A generating class for a graph, see examples below |
| x | A list containing a generating class for a graph, see examples below |
| forceCheck | Logical determining if it should be checked if the graph is acyclical |
| result | The format of the graph. The possible choices are "NEL" (for a graphNEL object), "matrix" (for an adjacency matrix), "igraph" (for an igraph object), "Matrix" (for a sparse matrix). |

## Value

Functions ug(), dag(), ugList() and dagList() return a 'graphNEL' object, an adjacency matrix or an 'igraph' object.

## Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

## Examples

```
ugr <- ug(~me:ve,~me:al,~ve:al,~al:an,~al:st,~an:st)

ugr <- ug(~me:ve:al,~al:an:st)

ugr <- ug(c("me","ve"),c("me","al"),c("ve","al"),c("al","an"),c("al","st"),c("an","st"))

ugr <- ug(~me:ve:al, c("me","ve"),c("me","al"),c("ve","al"),c("al","an"),c("al","st"),c("an","st"))


dagr <- dag(c("me","ve"),c("me","al"),c("ve","al"),c("al","an"),c("al","st"),c("an","st"))


dagr <- dag(~me:ve,~me:al,~ve:al,~al:an,~al:st,~an:st)

dagr <- dag(~me:ve:al,~ve:al:an)

graph::edges(ugr)
graph::nodes(ugr)

graph::edges(dagr)
graph::nodes(dagr)

ugList(list(~me:ve:al,~al:an:st))
dagList(list(~me:ve:al,~ve:al:an))
```

vpar *List of vertices and their parents for graph.*

## Description

Get list of vertices and their parents for graph.

## Usage

```
vpar(object, getv = TRUE, forceCheck = TRUE)
## S3 method for class 'graphNEL'
vpar(object, getv = TRUE, forceCheck = TRUE)
## S3 method for class 'matrix'
vpar(object, getv = TRUE, forceCheck = TRUE)
```

## Arguments

| | |
|---|---|
| object | An object representing a graph. Valid objects are an adjacency matrix or as a graphNEL. |
| getv | The result is by default a list of vectors of the form (v, pa1, pa2, ... paN) where pa1, pa2, ... paN are the parents of v. If getv is FALSE then the vectors will have the form (pa1, pa2, ... paN). |
| forceCheck | Logical indicating if it should be checked that the object is a DAG. |

## Value

A list of vectors where each vector will have the form (v, pa1, pa2, ... paN) where pa1, pa2, ... paN are the parents of v.

## Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

## See Also

dag ug

## Examples

```
## DAGs
dagMAT <- dag(~a:b:c+c:d:e, result="matrix")
dagNEL <- dag(~a:b:c+c:d:e, result="NEL")

vpar(dagMAT)
vpar(dagNEL)
vpar(dagMAT, getv=FALSE)
vpar(dagNEL, getv=FALSE)
```

```
## Undirected graphs
ugMAT <- ug(~a:b:c+c:d:e, result="matrix")
ugNEL <- ug(~a:b:c+c:d:e, result="NEL")

## Not run:
## This will fail because the adjacency matrix is symmetric and the
## graphNEL has undirected edges
vpar(ugMAT)
vpar(ugNEL)

## End(Not run)

## When forceCheck is FALSE, it will not be detected that the graphs are undirected.
vpar(ugMAT, forceCheck=FALSE)
vpar(ugNEL, forceCheck=FALSE)

## Bidirected graphs
## This is, for graphNEL's, the same as working with bidirected edges:
if (require(graph)){
graph::edgemode(ugNEL)
graph::edgemode(ugNEL) <- "directed"
graph::edgemode(ugNEL)
vpar(ugNEL,FALSE)
}
```

---

`wine` *Chemical composition of wine*

---

## Description

Using chemical analysis determine the origin of wines

## Usage

```
data(wine)
```

## Format

A data frame with 178 observations on the following 14 variables.

`Cult` a factor with levels `v1` `v2` `v3`: 3 different graph varieties

`Alch` Alcohol

`Mlca` Malic acid

`Ash` Ash

`Aloa` Alcalinity of ash

`Mgns` Magnesium

`Ttlp` Total phenols

`Flvn` Flavanoids

`Nnfp` Nonflavanoid phenols

`Prnt` Proanthocyanins

`Clri` Color intensity

`Hue` Hue

`Oodw` OD280/OD315 of diluted wines

`Prln` Proline

## Details

Data comes from the UCI Machine Learning Repository. The grape variety `Cult` is the class identifier.

## Source

Frank, A. & Asuncion, A. (2010). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

## References

See references at http://archive.ics.uci.edu/ml/datasets/Wine

## Examples

```
data(wine)
## maybe str(wine) ; plot(wine) ...
```

# Index