# Package 'lcd'

February 20, 2015

**Type** Package

**Title** Learn Chain graphs via Decomposition

**Version** 0.7-3

**Date** 2012-11-09

**Author** Zongming Ma and Xiangrui Meng

**Maintainer** Zongming Ma <zongming.ma@gmail.com>

**Depends** R(>= 2.12.0), igraph, methods

**Imports** MASS, ggm

**Description** Functions for learning chain graphs (and as a special case, Bayesian networks) via the decomposition approach.

**License** GPL (>= 2)

**LazyData** yes

**LazyLoad** yes

**Repository** CRAN

**Date/Publication** 2012-11-11 14:15:12

**NeedsCompilation** yes

## R topics documented:

---

`lcd-package` *Structural learning of chain graphs via the decomposition approach*

---

## Description

This package implements the algorithms for learning chain graphs (and as a special case, Bayesian networks) via the decomposition approach as described in Ma, Xie and Geng (2008)and Xie, Geng and Zhao (2006). The correctness of the algorithms is given in the above cited paper.

The algorithms are *constraint-based* method: the actual learning procedure depends on testing statistical significances. See the learning functions for the particular tests we used in the implementation.

Currently, the package supports learning with continuous and discrete data, but not the mix of them.

The package also provides some utility functions for graph manipulation, frequency table construction and compression and random distribution and random sample generation from chain graphs.

## List of functions

The package has the following categories of functions.

- Chain graph/Bayesian network learning: `learn.mec.norm`, `learn.mec.multinom`, `learn.skeleton.norm`, `learn.skeleton.multinom`, `learn.complex.norm`, `learn.complex.multinom`, `learn.v`.
- Graphical model manipulation and graph plotting: `draw`, `is.chaingraph`, `is.separated`, `moralize`, `pattern`, `skeleton`, `tri.ug`, `maxcard.search`, `ug.to.jtree`.
- Random distribution and random sample generation from chain graph: `get.normal.dist`, `get.multinom.dist`, `rnorm.cg`, `rmultinom.cg`.

- Conditional independence tests: norm.ci.test, multinom.ci.test.
- Construction of undirected independence graphs from data: naive.getug.norm, naive.getug.multinom.
- Frequency table manipulation: as.freq.tb, compress.freq.tb.
- Graph Comparison: comp.pat, comp.skel.

Moreover, the package include the following two graph structure:

- The ALARM network: alarm.net;
- A toy example: toy.graph.

## Author(s)

Zongming Ma
Department of Statistics
The Wharton School, University of Pennsylvania, USA
<zongming.ma@gmail.com>

Xiangrui Meng
Institute of Computational and Mathematical Engineering
Stanford University, USA

## References

Ma, Z., Xie, X. and Geng, Z. (2008). Structural learning of chain graphs via decomposition. *J. Mach. Learn. Res.,* **9**, 2847-2880.

Xie, X., Geng, Z. and Zhao, Q. (2006). Decomposition of structural learning about directed acyclic graphs. *Artif. Intell.,* **170**, 422-439.

## Examples

```
set.seed(100)
p.value <- .01
n <- 3000
is.chaingraph(toy.graph)
tgdata <- rnorm.cg(n, toy.graph, get.normal.dist(toy.graph))
tgug <- naive.getug.norm(tgdata, p.value)
tg.jtree <- ug.to.jtree(tgug)
tg.pat <- learn.mec.norm(tg.jtree, cov(tgdata), n, p.value, "CG")
comp.skel(skeleton(toy.graph), skeleton(tg.pat))
comp.pat(pattern(toy.graph), tg.pat)
```

## alarm.net *Graph structure of ALARM network*

### Description

The adjacency matrix of the ALARM network graph structure.

### Usage

```
data(alarm.net)
```

### Source

Beinlich, I., Suermondt, H., Chavez, R. and Cooper, G. (1989). The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks, in: *Proceedings of the 2nd European Conference on Artificial Intelligence in Medicine*, pp.247-256, Springer-Verlag, Berlin.

## all.equal-methods *All.equal method for sep.pair class*

### Description

Checks if two `sep.pair` objects are all.equal. We call two `sep.pair` objects all.equal if they have the same (u,v) set, regardless of order.

### Methods

**target = "sep.pair", current = "sep.pair"** Returns `TRUE` if `target` and `current` have the same (u,v) set.

## as.freq.tb *Frequency table transformation*

### Description

Transforms a discrete data matrix into a raw frequency table.

### Usage

```
as.freq.tb(mat)
```

### Arguments

| | |
|---|---|
| mat | a data matrix, with each row corresponding to an observation. |

### Details

Each element in the mat is expected to an integer or a factor.

### Value

An object of class freq.tb

### Author(s)

Zongming Ma and Xiangrui Meng

### See Also

compress.freq.tb.

---

## comp.pat *Pattern comparison*

---

### Description

Compares a (learned) chain graph pattern to the (supposed) true pattern. The two patterns should have the same vertex set in order for the function to return a meaningful result.

### Usage

```
comp.pat(truepat, pat)
```

### Arguments

| | |
|---|---|
| truepat | the adjacency matrix of the true pattern. |
| pat | the adjacency matrix of the pattern to be compared with the true one. |

### Value

| | |
|---|---|
| a.total | total number of complex arrows on the true pattern. |
| a.missing | number of true complex arrows missing in the pattern to be compared. |
| a.extra | number of spurious complex arrows present in the pattern to be compared. |
| shd | structural Hamming distance from pat to truepat. |

### Note

Structural Hamming distance is defined as the total number of operations needed to convert one graph to the other. Each operation must be one of the following: (1) add or delete an undirected edge, or (2) add, remove or reverse an orientation of an edge.

## Author(s)

Zongming Ma and Xiangrui Meng

## References

Tsamardinos, I., Brown, L., and Aliferis, C. (2006). The max-min hill-climbing Bayesian network structure learning algorithm. *Mach. Learn.*, **65**(1):31-78.

## See Also

comp.skel.

---

# comp.skel *Skeleton comparison*

---

## Description

Compares a graph skeleton to the (supposed) true skeleton. The two skeletons should have the same set of vertices in order for the function to return a meaningful result.

## Usage

```
comp.skel(trueskel, skel)
```

## Arguments

| | |
|---|---|
| trueskel | the adjacency matrix of the true skeleton. |
| skel | the adjacency matrix of the skeleton to be compared with the true one. |

## Value

| | |
|---|---|
| e.total | total number of edges on the true skeleton. |
| e.missing | number of true edges missing in the skeleton to be compared. |
| e.extra | number of spurious edges present in the skeleton to be compared. |

## See Also

comp.pat.

## compress.freq.tb *Frequency table compression*

### Description

Compresses a frequency table to a subset of the variables.

### Usage

```
compress.freq.tb(tb, subset = colnames(tb@table)[-ncol(tb@table)])
```

### Arguments

| | |
|---|---|
| tb | the frequency table to be compressed, an object of class freq.tb. |
| subset | a vector of strings indicating the subset of variables to be compressed on. The default is the whole variable set. |

### Details

The actual computation routine is implemented in C++.

### Value

The compressed table, an object of class freq.tb.

### Author(s)

Zongming Ma and Xiangrui Meng

## draw *Draw graph*

### Description

Draws a graph from its adjacency matrix.

### Usage

```
draw(amat, plain = TRUE)
```

### Arguments

| | |
|---|---|
| amat | the adjacency matrix of the graph. |
| plain | logical value. If TRUE, then a plain plot is drawn for the graph; if FALSE, a tlkck plot is drawn. |

### Details

The function converts the adjacency matrix to an `igraph` object and uses the `plot.igraph`/`tkplot` function in `igraph` package to draw the graph. Special care is paid to make the directed/undirected edges displayed right. Finally, the function returns the vertex list of the graph for comparison to the vertex numbering used in the plot.

### Value

The vertex list. The main output is the side effect of the function.

### Author(s)

Zongming Ma and Xiangrui Meng

---

## `freq.tb-class` *Class "freq.tb"*

---

### Description

Objects representing frequency counts of all configurations of discrete variables existing in a data set.

### Objects from the Class

Objects can be created by calls of the form `new("freq.tb", ...)`.

### Slots

**`table`:** an $m$ by $p+1$ matrix, where in each row, the first $p$ integers give a configuration and the $p+1$-th element records the frequency count.

**`levels`:** a $p$-vector which gives the total levels for each variable.

### Warning

The user is responsible for giving the correct `levels`!

### Author(s)

Zongming Ma and Xiangrui Meng

### Examples

```
showClass("freq.tb")
```

`get.multinom.dist` *Random multinomial distribution generation from a chain graph*

## Description

Generates a random multinomial distribution from a given chain graph structure.

## Usage

```
get.multinom.dist(amat, n.state, alpha, beta)
```

## Arguments

| | |
|---|---|
| `amat` | the adjacency matrix of a chain graph. |
| `n.state` | a vector of positive integers indicating the desired number of states for each variable in the chain graph. |
| `alpha, beta` | shape parameters for Beta distribution in simulating the potentials. |

## Details

The function returns a list that encodes a block-recursive conditional distribution over the chain graph structure.

## Value

A sequence of block-recursive conditional distributions generated according to the graph structure.

## Author(s)

Zongming Ma and Xiangrui Meng

## References

Cowell, R. G., Dawid, A. P., Lauritzen, S. L. and Spiegelhalter, D. J. (1999) *Probabilistic Networks and Expert Systems*. pp.77-79. Springer-Verlag, New York.

## get.normal.dist *Random normal distribution generation from a chain graph*

### Description

Generates a random normal distribution from a given chain graph structure.

### Usage

```
get.normal.dist(amat)
```

### Arguments

| | |
|---|---|
| amat | the adjacency matrix of a chain graph. |

### Details

The function returns a matrix that encodes a mean 0 block-recursive regression model (See Wermuth (1992)).

### Value

A matrix encoding the randomly generated block-recursive regression model.

### Author(s)

Zongming Ma and Xiangrui Meng

### References

Wermuth, N. (1992). Block-recursive regression equations (with discussions). *Revista Brasileira de Probabilidade e Estatistica,* **6**, 1-56.

## is.chaingraph *Chain graph verification*

### Description

Checks if a given graph is a chain graph.

### Usage

```
is.chaingraph(amat)
```

### Arguments

| | |
|---|---|
| amat | the adjacency matrix of the graph with dimnames. |

## Value

| | |
|---|---|
| result | a logical value, TRUE if the given graph is a valid chain graph and FALSE otherwise. |
| vert.order | a topological order of the graph vertices. |
| chain.size | a vector indicating the size of each chain component in the graph, whose order corresponds to vert.order. |

## Author(s)

Zongming Ma and Xiangrui Meng

## References

Cowell, R. G., Dawid, A. P., Lauritzen, S. L. and Spiegelhalter, D. J. (1999) *Probabilistic Networks and Expert Systems*. Springer-Verlag, New York.

## Examples

```
data(lcd)
is.chaingraph(toy.graph)
```

---

# is.separated *c-separation on the chain graph*

---

## Description

Checks whether two vertices u and v are $c$-separated by a set sep on the given chain graph.

## Usage

```
is.separated(u, v, sep, amat)
```

## Arguments

| | |
|---|---|
| u | one vertex under investigation. |
| v | the other vertex under investigation. |
| sep | the candidate separator. |
| amat | the adjacency matrix of a chain graph. |

## Details

The actual function uses the moralization criterion instead of the separation criterion. They are equivalent and the former is easier to implement.

## Value

A boolean value indicating whether separation holds or not.

## Author(s)

Zongming Ma and Xiangrui Meng

## References

Lauritzen, S. L. (1996). *Graphical Models.* Clarendon Press, Oxford.

Studeny, M. and Bouckaert, R. R. (1998). On chain graph models for description of conditional independence structures. *Annals of Statistics* **26** 1434-1495.

## Examples

```
is.separated("a", "d", c("b","c"), toy.graph)
```

---

`learn.graph` *Graph learning functions*

---

## Description

Learns chain graphs structures via decomposition algorithms with normal or multinomial data.

## Usage

```
learn.complex.norm(skel, cov, n, p.value)

learn.mec.norm(tree, cov, n, p.value, method = "CG")

learn.skeleton.norm(tree, cov, n, p.value, drop = TRUE)

learn.complex.multinom(skel, freq.tb, p.value)

learn.mec.multinom(tree, freq.tb, p.value, method = "CG")

learn.skeleton.multinom(tree, freq.tb, p.value, drop = TRUE)

learn.v(skel, tree)
```

## Arguments

| | |
|---|---|
| `skel` | the object returned by `learn.skeleton.norm` function. |
| `tree` | an object of class `sep.tree`, e.g., the separation tree obtained via `ug.to.jtree`. |
| `cov` | the covariance matrix of the data |
| `n` | number of random samples used to obtain the covariance matrix. |
| `freq.tb` | the frequency table of the data, should be an object from class `freq.tb`. |
| `p.value` | thresholding $p$-value for conditional independence test during the learning procedure. |

| | |
|---|---|
| method | a character string, must be one of "CG", "DAG" and "UG", telling the alogrithm whether the underlying graph is a DAG, undirected graph or more generally a chain graph. |
| drop | logical value, with default=TRUE, whether to drop possibly extra edges after recovering and combining local skeletons. |

## Details

learn.mec.norm is the wrapper function for learn.skeleton.norm, learn.complex.norm and learn.v for normal data, which is intended to be mostly called by the user. learn.mec.multinom is the counterpart for discrete data. The user specifies whether she/he is learning an undirected graph (Markov network), a DAG (Bayesian network) or in general a chain graph by supplying the specific string to the method argument. The function itself will utilize the correct helper functions to perform the structural learning and return the pattern (representing the Markov equivalent class) of the graph.

learn.skeleton.norm and learn.skeleton.multinom try to learn the skeleton of the graph. learn.v, learn.complex.norm (or learn.complex.multinom) are for v-structure/complex discovery in DAG/chain graph respectively.

Some parts of the code of these learning functions are adapted from the code for pcAlgo function in the R package pcalg written by Markus Kalisch and Martin Maechler.

## Value

learn.mec.norm, learn.mec.multinom, learn.v, learn.complex.norm, learn.complex.multinom return the pattern of the graph, represented in its adjacency matrix.

learn.skeleton.norm and learn.skeleton.multinom return a list of the following two items:

| | |
|---|---|
| amat | the adjacency matrix of the learned skeleton. |
| sep.pairs | a list of sep.pair objects, recording the separation pairs we obtained during the skeleton learning procedure. |

## Author(s)

Zongming Ma and Xiangrui Meng

## References

Ma, Z., Xie, X. and Geng, Z. (2008). Structural learning of chain graphs via decomposition. *J. Mach. Learn. Res.,* **9**, 2847-2880.

Xie, X., Geng, Z. and Zhao, Q. (2006). Decomposition of structural learning about directed acyclic graphs. *Artif. Intell.,* **170**, 422-439.

## Examples

```
set.seed(100)
p.value <- .01
n <- 3000
is.chaingraph(toy.graph)
```

```
tgdata <- rnorm.cg(n, toy.graph, get.normal.dist(toy.graph))
tgug <- naive.getug.norm(tgdata, p.value)
tg.jtree <- ug.to.jtree(tgug)
tg.pat <- learn.mec.norm(tg.jtree, cov(tgdata), n, p.value, "CG")
comp.skel(skeleton(toy.graph), skeleton(tg.pat))
comp.pat(pattern(toy.graph), tg.pat)
```

---

`maxcard.search` *Maximum cardinality search*

---

## Description

Performs a maximum cardinality search on an undirected graph to determine whether it is triangulated.

## Usage

```
maxcard.search(amat)
```

## Arguments

| | |
|---|---|
| `amat` | the adjacency matrix of the undirected graph. |

## Value

| | |
|---|---|
| `is.triangulated` | a logical value indicating whether the input graph is triangulated or not. |
| `perfect.numbering` | a perfect numbering of the vertices. |
| `card` | number of unlabeled neighbors when labeling each variable, with order compatible to the perfect numbering. |
| `pi.record` | a record of unlabeled neighbors during the execution of the algorithm. |

## Note

Only the `is.triangulated` and `perfect.numbering` are supposed to be of interest to the users. The other two output components are mainly for track purpose.

## Author(s)

Zongming Ma and Xiangrui Meng

## References

Tarjan, R. E. and Yannakakis, M. (1984). Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM Journal on Computing*, **13**, 566-79.

## moralize *Chain graph moralization*

### Description

Computes the moral graph of a given chain graph.

### Usage

```
moralize(amat)
```

### Arguments

| | |
|---|---|
| amat | the adjacency matrix of the chain graph. |

### Details

Joins the parents of every complex and undirect all the edges.

### Value

The adjacency matrix of the moral graph.

### Author(s)

Zongming Ma and Xiangrui Meng

### References

Lauritzen, S. L. (1996). *Graphical Models.* Clarendon Press, Oxford.

### Examples

```
moralize(toy.graph)
```

## multinom.ci.test *Conditional independence test for multinomial data*

### Description

Performs a conditional independence test for variables u and v with conditioning set cond.

### Usage

```
multinom.ci.test(tb, u, v, cond = c())
```

### Arguments

| | |
|---|---|
| tb | a frequency table summarizing the data, an object of class freq.tb. |
| u | name of one variable under investigation. |
| v | name of the other variable under investigation. |
| cond | a vector of variable names indicating the conditioning set in the test. |

### Details

The test is based on the deviance test in log-linear model for count data. The null hypothesis is that all interaction terms involving both u and v vanish.

### Value

| | |
|---|---|
| deviance | the deviance of the saturated model from the model under null hypothesis. |
| df | the degrees of freedom for the deviance. |
| p.value | the approximate $p$-value for the test based on Chi-square approximation. |

### Author(s)

Zongming Ma and Xiangrui Meng

### References

Agresti, A. (2002). *Categorical Data Analysis.* 2nd Ed. John Wiley and Sons. Hoboken, NJ.

---

## naive.getug.multinom *A naive function to get an undirected graph for multinomial data*

---

### Description

Learns an undirected independence graph from a given data set. The data are assumed to be multinomially distributed.

### Usage

```
naive.getug.multinom(freq.tb, p.value, method = "mkb")
```

### Arguments

| | |
|---|---|
| freq.tb | an object of class freq.tb, the frequency table for the data. |
| p.value | the thresholding $p$-value for each conditional independence test. |
| method | a string to specify a method, see details. |

### Details

Currently, there are three supported `method`'s. `"mkb"` stands for grow-shrink Markov blanket selection, where greedy Markov blanket selection is performed for each vertex and the blankets are the neighborhood for vertices of the output graph. The `"simple"` method simply performs conditional independence test for each vertex pair with all the rest vertices as the conditioning set. The `"fwd"` method uses the forward selection procedure described in Edwards (2000).

### Value

The adjacency matrix of an undirected graph.

### Author(s)

Zongming Ma and Xiangrui Meng

### References

Edwards, D. (2000). *Introduction to Graphical Modelling.* 2nd Ed. Springer-Verlag, New York.

---

## `naive.getug.norm` *A naive function to get an undirected graph for normal data*

---

### Description

Learns an undirected independence graph from a given data set. The data are assumed to be normally distributed.

### Usage

```
naive.getug.norm(data, p.value)
```

### Arguments

| | |
|---|---|
| `data` | a data matrix with rows corresponding to observations and columns corresponding to random variables. |
| `p.value` | the thresholding $p$-value for each conditional independence test. |

### Details

For each pair of random variables, the function performs a conditional independence using the partial correlation coefficient. If the p-value of the test is smaller than the given threshold, then there will be an edge on the output graph. The function essentially uses the global Markov property of the undirected graph.

### Value

The adjacency matrix of the computed undirected graph, with dimnames the corresponding random variables.

### Author(s)

Zongming Ma and Xiangrui Meng

### References

Lauritzen, S. L. (1996). *Graphical Models.* Clarendon Press, Oxford.

---

## norm.ci.test *Conditional independence test for multivariate normal data*

---

### Description

Performs a conditional independence test for multivariate normal variables u and v with conditioning set cond.

### Usage

```
norm.ci.test(cov, n, u, v, cond = c())
```

### Arguments

| | |
|---|---|
| cov | the sample covariance matrix. |
| n | number of samples used to compute cov. |
| u | name of one variable under investigation. |
| v | name of the other variable under investigation. |
| cond | a vector of variable names indicating the conditioning set in the test. |

### Details

The function performs a likelihood ratio test as described in Whittaker (1990), Chapter 6.

### Value

| | |
|---|---|
| deviance | the Chisq-statistic computed under the null hypothesis. |
| df | the degrees of freedom for the Chisq-statistic. |
| p.value | the $p$-value for the test. |

### Note

Some of the C routines used were originally written by Robert Castelo <robert.castelo@upf.edu>, Alberto Roverato <alberto.roverato@unibo.it> in the qp package. The authors added the part for calculate the likelihood ratio test-statistic and modified the R wrapper function.

### Author(s)

Zongming Ma and Xiangrui Meng

## References

Whittaker, J. (1990). *Graphical Models in Applied Mathematical Multivariate Statistics.* John Wiley and Sons, Chichester, England.

---

`pattern` *Chain graph pattern computation*

---

## Description

Extracts the pattern of a chain graph.

## Usage

```
pattern(amat)
```

## Arguments

| | |
|---|---|
| `amat` | the adjacency matrix of a chain graph. |

## Value

The adjacency matrix of the chain graph pattern.

## Author(s)

Zongming Ma and Xiangrui Meng

## References

Studeny, M. (1997). A recovery algorithm for chain graphs. *Int. J. Approx. Reasoning,* **17**, 265-293.

---

`rmultinom.cg` *Random multinomial sample from a chain graph*

---

## Description

Generates a desired number of multinomial random samples from a given chain graph structure and a sequence of compatible block-recursive conditional distributions.

## Usage

```
rmultinom.cg(n, amat, distn)
```

### Arguments

| | |
|---|---|
| n | the intended sample size, should be at least 1. |
| amat | the adjacency matrix of a chain graph. |
| distn | a sequence of block-recursive conditional distributions, e.g. the one returned by get.multinom.dist. |

### Value

An $n$ by $p$ matrix with each row corresponding to a random sample.

### Author(s)

Zongming Ma and Xiangrui Meng

### References

Cowell, R. G., Dawid, A. P., Lauritzen, S. L. and Spiegelhalter, D. J. (1999) *Probabilistic Networks and Expert Systems*. pp.77-79. Springer-Verlag, New York.

---

## rnorm.cg *Random normal sample from a chain graph*

---

### Description

Generates a desired number of normal random samples from a given chain graph structure and a given block-recursive regression system compatible with the chain graph.

### Usage

```
rnorm.cg(n, amat, Bstar)
```

### Arguments

| | |
|---|---|
| n | the intended sample size, should be at least 1. |
| amat | the adjacency matrix of a chain graph. |
| Bstar | the matrix representation of the block-recursive regression structure, e.g., the one returned by get.normal.dist. |

### Details

The function uses a mean 0 block-recursive regression model (See Wermuth (1992)), which is recorded in Bstar and normal random samples are generated from the specified block-recursive regression model.

### Value

An $n$ by $p$ matrix with each row corresponding to a random sample.

## Author(s)

Zongming Ma and Xiangrui Meng

## References

Wermuth, N. (1992). Block-recursive regression equations (with discussions). *Revista Brasileira de Probabilidade e Estatistica,* **6**, 1-56.

---

`sep.pair-class` *Class "sep.pair"*

---

## Description

Objects representing separation pairs in a graphical model, with slots u and v for the pair of vertices and slot s for their separator, which indicates the relation u and v are separated by s (on some graph $G$).

## Objects from the Class

Objects can be created by calls of the form `new("sep.pair", ...)`.

## Slots

**u, v:** Objects of class `"character"`, containing the pair of vertices.

**s:** Object of class `"character"`, containing the separator of u and v.

## Methods

**all.equal** `signature(target = "sep.pair", current = "sep.pair")`: see `all.equal`.

**show** `signature(object = "sep.pair")`.

## Examples

```
showClass("sep.pair")
```

## sep.tree-class *Class "sep.tree"*

### Description

Objects representing separation tree as described in Xie, Geng and Zhao (2006) and Ma, Xie and Geng (2008), which includes junction tree of cliques as a special case.

### Objects from the Class

Objects can be created by calls of the form `new("sep.tree", ...)`. We recommend user to use the functions provided in this package, such as `ug.to.jtree` to create the separation tree object.

### Slots

`tree.struct` the adjacency matrix of the junction tree.

`cliques` the list of cliques on the junction tree.

`separators` the list of separators on the junction tree, each element on the list has two components: the `separator` compoenent is the set of graph vertices in the separator and `edge` is the edge on the tree that the separator is attahced to.

### Author(s)

Zongming Ma and Xiangrui Meng

### References

Ma, Z., Xie, X. and Geng, Z. (2008). Structural learning of chain graphs via decomposition. *J. Mach. Learn. Res.,* **9**, 2847-2880.

Xie, X., Geng, Z. and Zhao, Q. (2006). Decomposition of structural learning about directed acyclic graphs. *Artif. Intell.,* **170**, 422-439.

### Examples

```
showClass("sep.tree")
```

## show-methods *Show method for sep.pair class*

### Description

Method for showing a `sep.pair` object.

### Methods

**object = "sep.pair"** Successively prints out the pairs (`u`, `v`) and their separator `s`.

## skeleton *Graph skeleton*

### Description

Returns the skeleton of a graph. The graph can be undirected, directed or mixed.

### Usage

```
skeleton(amat)
```

### Arguments

| | |
|---|---|
| amat | the adjacency matrix of the graph. |

### Value

The adjacency matrix of the graph skeleton.

### Author(s)

Zongming Ma and Xiangrui Meng

### Examples

```
skeleton(toy.graph)
```

## toy.graph *Graph structure of a toy example*

### Description

The adjacency matrix of a toy chain graph example.

### Usage

```
data(toy.graph)
```

## tri.ug *Triangulation of an undirected graph*

### Description

Triangulates an undirected graph to a chordal graph.

### Usage

```
tri.ug(amat)
```

### Arguments

| | |
|---|---|
| amat | the adjacency matrix of undirected graph. |

### Details

The function implements the 'One-step look ahead triangulation' algorithm described in Cowell, et al (1999). The criterion $c(v)$ is chosen to be the number of edges needed to be added to the graph if vertex $v$ is chosen to be labelled.

### Value

The adjacency matrix of the triangulated graph.

### Author(s)

Zongming Ma and Xiangrui Meng

### References

Cowell, R. G., Dawid, A. P., Lauritzen, S. L. and Spiegelhalter, D. J. (1999) *Probabilistic Networks and Expert Systems*. Springer-Verlag, New York.

## ug.to.jtree *Junction tree construction for undirected graph*

### Description

Constructs a junction tree for an undirected graph.

### Usage

```
ug.to.jtree(ugamat)
```

## Arguments

| | |
|---|---|
| ugamat | the adjacency matrix of the undirected graph. |

## Details

This function and its helpers implement the junction tree construction algorithm described in detail in Section 4.3 and 4.4 of Cowell, et al (1999).

## Value

An object of class sep.tree.

## Author(s)

Zongming Ma and Xiangrui Meng

## References

Cowell, R. G., Dawid, A. P., Lauritzen, S. L. and Spiegelhalter, D. J. (1999) *Probabilistic Networks and Expert Systems*. Springer-Verlag, New York.

## See Also

sep.tree-class.

# Index