**Description**

**CGstats** provides what corresponds to calling `cow wt` on different strata of data where the strata are defined by the combinations of factors in data.

**Usage**

```r
CGstats(object, varnames = NULL, homogeneous = TRUE, simplify = TRUE)
```

### Arguments

- `object`
- `varnames`
- `homogeneous`
- `simplify`

**Value**

A list whose form depends on the type of input data and the varnames.

**Author(s)**

Søren Højsgaard, <sorenh@math.aau.dk>

**See Also**

`cov wt`
ciTest

Examples

data(milkcomp)

CGstats(milkcomp)
CGstats(milkcomp, c(1,2))
CGstats(milkcomp, c("lactime","treat"))
CGstats(milkcomp, c(3,4))
CGstats(milkcomp, c("fat","protein"))
CGstats(milkcomp, c(2,3,4), simplify=FALSE)
CGstats(milkcomp, c(2,3,4), homogeneous=FALSE)
CGstats(milkcomp, c(2,3,4), simplify=FALSE, homogeneous=FALSE)

ciTest

Generic function for conditional independence test

Description

Generic function for conditional independence test. Specializes to specific types of data.

Usage

   ciTest(x, set=NULL, ...)

   ## S3 method for class 'data.frame'
   ciTest(x, set=NULL, ...)

   ## S3 method for class 'table'
   ciTest(x, set=NULL, ...)

   ## S3 method for class 'list'
   ciTest(x, set=NULL, ...)

Arguments

   x     An object for which a test for conditional independence is to be made. See 'details' for valid types of x.

   set   A specification of the test to be made. The tests are of the form u and v are independent conditionally on S where u and v are variables and S is a set of variables. See 'details' for details about specification of set.

   ...   Additional arguments to be passed on to other methods.
Details

x can be 1) a table, 2) a dataframe whose columns are numerics and factors or 3) a list with components cov and n.obs.

set can be 1) a vector or 2) a right-hand sided formula in which variables are separated by ‘+’. In either case, it is tested if the first two variables in the set are conditionally independent given the remaining variables in set. (Notice an abuse of the ‘+’ operator in the right-hand sided formula: The order of the variables does matter.)

Value

An object of class 'citest' (which is a list).

Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

See Also

ciTest.table ciTest_table
ciTest.data.frame ciTest_df
ciTest.list ciTest_mvn
chiq.test

Examples

## contingency table:
data(reinis)
## dataframe with only numeric variables:
data(carcass)
## dataframe with numeric variables and factors:
data(milkcomp1)

ciTest(cov.wt(carcass, method='ML'), set=-Fat11+Meat11+Fat12)
ciTest(reinis, set=-smo+phy+sys)
ciTest(milkcomp1, set=-tre+fat+pro)

---

Test for conditional independence in a dataframe

Description

Test for conditional independence in a dataframe.
ciTest_df

Usage

ciTest_df(x, set = NULL, ...)

Arguments

x A dataframe.
set
...

Details

set can be 1) a vector or 2) a right-hand sided formula in which variables are separated by '+'.
In either case, it is tested if the first two variables in the set are conditionally independent given
the remaining variables in set. (Notice an abuse of the '+' operator in the right-hand sided formula:
The order of the variables does matter.)
If set is NULL then it is tested whether the first two variables are conditionally independent given
the remaining variables.
If set consists only of factors then x[, set] is converted to a contingency table and the test is made
in this table using ciTest_table().
If set consists only of numeric values and integers then x[, set] is converted to a list with com-
ponents cov and n.obs by calling cov.wt(x[, set], method='ML'). This list is then passed on to
ciTest_mvn() which makes the test.

Value

An object of class 'citest' (which is a list).

Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

See Also

ciTest
   ciTest_table
   ciTest_table
   ciTest_list
   ciTest_mvn
   chisq.test

Examples

data(milkcomp1)
ciTest(milkcomp1, set=-tre+fat+pro)
ciTest_df(milkcomp1, set=-tre+fat+pro)
ciTest_mvn  

*Test for conditional independence in the multivariate normal distribution*

Description
Test for conditional independence in the multivariate normal distribution.

Usage
```r
ciTest_mvn(x, set = NULL, statistic = "DEV", ...)
```

Arguments
- **x**: A list with elements `cov` and `nNobs` (such as returned from calling `cov.wt()` on a dataframe. See examples below.)
- **set**:
- **statistic**: `NULL`
- **...**

Details
- `set` can be 1) a vector or 2) a right-hand sided formula in which variables are separated by `'+'`. In either case, it is tested if the first two variables in the `set` are conditionally independent given the remaining variables in `set`. (Notice an abuse of the `'+` operator in the right-hand sided formula: The order of the variables does matter.)
- If `set` is `NULL` then it is tested whether the first two variables are conditionally independent given the remaining variables.
- `x` must be a list with components `cov` and `nNobs` such as returned by calling `cov.wt()`, `method='ML'` on a dataframe.

Value
An object of class `citest` (which is a list).

Author(s)
Søren Højsgaard, <sorenh@math.aau.dk>

See Also
- `ciTest`
- `ciTest.table`
- `ciTest_table`
- `ciTest.data.frame`
- `ciTest_df`
- `chisq.test`
ciTest_ordinal

A function to compute Monte Carlo and asymptotic tests of conditional independence for ordinal and/or nominal variables.

Description

The function computes tests of independence of two variables, say u and v, given a set of variables, say S. The deviance, Wilcoxon, Kruskal-Wallis and Jonckheere-Terpstra tests are supported. Asymptotic and Monte Carlo p-values are computed.

Usage

ciTTest_ordinal(x, set = NULL, statistic = "dev", N = 0, ...)

Arguments

x
A dataframe or table.

set
The variable set (u,v,S), given either as an integer vector of the column numbers of a dataframe or dimension numbers of a table, or as a character vector with the corresponding variable or dimension names.

statistic
Either "deviance", "wilcoxon", "kruskal" or "jt".

N
The number of Monte Carlo samples. If N<=0 then Monte Carlo p-values are not computed.

... Additional arguments, currently not used

Details

The deviance test is appropriate when u and v are nominal; Wilcoxon, when u is binary and v is ordinal; Kruskal-Wallis, when u is nominal and v is ordinal; Jonckheere-Terpstra, when both u and v are ordinal.

Value

A list including the test statistic, the asymptotic p-value and, when computed, the Monte Carlo p-value.

P
Asymptotic p-value

montecarlo.P
Monte Carlo p-value

Author(s)

Flaminia Musella, David Edwards, Søren Højsgaard, <sorenh@math.aau.dk>
References


See Also
ciTest_table, ciTest

Examples

```r
library(grim)
data(dumping, package="gRbase")

ciTTest_ordinal(dumping, c(2,1,3), stat="jt", N=1000)
ciTTest_ordinal(dumping, c("Operation","Symptom","Centre"), stat="jt", N=1000)
ciTTest_ordinal(dumping, ~ Operation + Symptom + Centre, stat="jt", N=1000)

data(reinis)
ciTTest_ordinal(reinis, c(1,3,4:6),N=1000)

# If data is a dataframe
dd <- as.data.frame(dumping)
ncells <- prod(dim(dd))
ff <- dd$Freq
idx <- unlist(mapply(function(i,n) rep(i,n),1:ncells,ff))
dumpDF <- dd[idx, 1:3]
rownames(dumpDF) <- 1:NROW(dumpDF)

ciTTest_ordinal(dumpDF, c(2,1,3), stat="jt", N=1000)
ciTTest_ordinal(dumpDF, c("Operation","Symptom","Centre"), stat="jt", N=1000)
ciTTest_ordinal(dumpDF, ~ Operation + Symptom + Centre, stat="jt", N=1000)
```

---

ciTTest_table

*Test for conditional independence in a contingency table*

Description

Test for conditional independence in a contingency table

Usage

```r
ciTTest_table(x, set = NULL, statistic = "dev", method = "chisq", adjust.df = TRUE, slice.info = TRUE, L = 20, B = 200, ...)
```
**Arguments**

- `x`  
  A contingency table.
- `set`  
- `statistic`  
- `method`  
- `adjust.df`  
- `slice.info`  
- `L`  
- `B`  
- `...`  
  Additional arguments.

**Details**

`set` can be 1) a vector or 2) a right-hand sided formula in which variables are separated by `'+'`. In either case, it is tested if the first two variables in the `set` are conditionally independent given the remaining variables in `set`. (Notice an abuse of the `'+` operator in the right-hand sided formula: The order of the variables does matter.)

If `set` is `NULL` then it is tested whether the first two variables are conditionally independent given the remaining variables.

**Value**

An object of class `citest` (which is a list).

**Author(s)**

Søren Højsgaard, <sorenh@math.aau.dk>

**See Also**

- `ciTest`
- `ciTest.data.frame`
- `ciTest_df`
- `ciTest.list`
- `ciTest_mvn`
- `chisq.test`

**Examples**

```r
data(reinis)
ciTest(reinis, set="smo+phy+sys")
ciTest_table(reinis, set="smo+phy+sys")
```
cmod

Graphical Gaussian model

Description

Specification of graphical Gaussian model. The ‘c’ in the name cmod refers to that it is a (graphical) model for ‘c’ontinuous variables

Usage

cmod(formula, data, marginal = NULL, fit = TRUE, details=0)

Arguments

formula Model specification in one of the following forms: 1) a right-hand sided formula, 2) as a list of generators, 3) an undirected graph (represented either as a graphNEL object or as an adjacency matrix). Notice that there are certain model specification shortcuts, see Section ‘details’ below

data Data in one of the following forms: 1) A dataframe or 2) a list with elements cov and n.obs (such as returned by the cov.wt() function.)
marginal Should only a subset of the variables be used in connection with the model specification shortcuts
fit Should the model be fitted.
details Control the amount of output; for debugging purposes.

Details

The independence model can be specified as ~.^1 and the saturated model as ~.^.. The marginal argument can be used for specifying the independence or saturated models for only a subset of the variables.

Value

An object of class cModel (a list)

Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

See Also

dmod mmod ggmfit
Examples

```r
## Graphical Gaussian model
data(carcass)
cm1 <- dmod(~ .^1, carcass)

## Stepwise selection based on BIC
cm2 <- backward(cm1, k = log(nrow(carcass)))

## Stepwise selection with fixed edges
cm3 <- backward(cm1, k = log(nrow(carcass)),
```

**dmod**  
*Log–linear model*

**Description**  
Specification of log–linear (graphical) model. The ‘d’ in the name `dmod` refers to that it is a (graphical) model for ‘discrete variables

**Usage**  
`dmod(formula, data, marginal, interactions=NULL, fit = TRUE, details=0)`

**Arguments**

- `formula`  
  Model specification in one of the following forms: 1) a right-hand sided formula, 2) as a list of generators, 3) an undirected graph (represented either as a graphNEL object or as an adjacency matrix). Notice that there are certain model specification shortcuts, see Section ‘details’ below.

- `data`  
  Either a table or a dataframe. In the latter case, the dataframe will be coerced to a table. See ‘details’ below.

- `interactions`  
  A number given the highest order interactions in the model, see Section ‘details’ below.

- `marginal`  
  Should only a subset of the variables be used in connection with the model specification shortcuts

- `fit`  
  Should the model be fitted.

- `details`  
  Control the amount of output; for debugging purposes.

**Details**  
The independence model can be specified as `~ .^1` and the saturated model as `~ .^..` Setting e.g. `interactions=3` implies that there will be at most three factor interactions in the model.

Data can be specified as a table of counts or as a dataframe. If data is a dataframe then it will be converted to a table (using `xtabs()`). This means that if the dataframe contains numeric values
then the you can get a very sparse and high dimensional table. When a dataframe contains numeric values it may be worthwhile to discretize data using the `cut()` function.

The marginal argument can be used for specifying the independence or saturated models for only a subset of the variables. When marginal is given the corresponding marginal table of data is formed and used in the analysis (notice that this is different from the behaviour of `loglin()` which uses the full table.

**Value**

An object of class `dModel`

**Author(s)**

Søren Højsgaard, <sorenh@math.aau.dk>

**See Also**

`cmod` `mmod`

**Examples**

```r
## Graphical log-linear model
data(reinis)
dm1<-dmod(~., reinis)
dm2<backward(dm1, k=2)
dm3<backward(dm1, k=2, fixin=list(c("family","phys","systol")))
## At most 3-factor interactions
dm1<-dmod(~., data=reinis, interactions=3)
```

---

**Description**

Setting formal classes for dModel, cModel and mModel objects

**Objects from the Class**

A virtual Class: No objects may be created from it.

**Slots**

`.S3Class`: Object of class "character" ~-

**Extends**

Class "cModel", directly. Class "mModel", by class "cModel", distance 2. Class "oldClass", by class "cModel", distance 3.
Methods

No methods defined with class "dModel" in the signature.

Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

Examples

showClass("dModel")

effloglin

Fitting Log-Linear Models by Message Passing

Description

Fit log-linear models to multidimensional contingency tables by Iterative Proportional Fitting.

Usage

effloglin(table, margin, fit = FALSE, eps = 0.01, iter=20, print = TRUE)

Arguments

table A contingency table
margin A generating class for a hierarchical log-linear model
fit If TRUE, the fitted values are returned.
eps Convergence limit; see 'details' below.
iter Maximum number of iterations allowed
print If TRUE, iteration details are printed.

Details

The function differs from loglin in that 1) data can be given in the form of a list of sufficient marginals and 2) the model is fitted only on the cliques of the triangulated interaction graph of the model. This means that the full table is not fitted, which means that effloglin is efficient (in terms of storage requirements). However effloglin is implemented entirely in R and is therefore slower than loglin.

Value

A list with components

comp1 Description of 'comp1'
comp2 Description of 'comp2'
...

getEdges

Find edges in a graph and edges not in an undirected graph.

Description

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

Usage

getEdges(object, type = "unrestricted", ingraph=TRUE, discrete=NULL, ...)
## S3 method for class 'list'
getEdges(object, type = "unrestricted", ingraph=TRUE, discrete=NULL, ...)
## S3 method for class 'graphNEL'
getEdges(object, type = "unrestricted", ingraph=TRUE, discrete=NULL, ...)
## S3 method for class 'matrix'
getEdges(object, type = "unrestricted", ingraph=TRUE, discrete=NULL, ...)

getInEdges(object, type = "unrestricted", discrete=NULL, ...)
getOutEdges(object, type = "unrestricted", discrete=NULL, ...)
getInEdgesMAT(adjmat, type = "unrestricted", discrete=NULL, ...)
getOutEdgesMAT(adjmat, type = "unrestricted", discrete=NULL, ...)

Arguments

object An object representing a graph; either a generator list, a graphNEL object or an adjacency matrix.
type Either "unrestricted" or "decomposable"
getEdges

**ingraph**
If TRUE the result is the edges in the graph; if FALSE the result is the edges not in the graph.

**discrete**
This argument is relevant only if object specifies a marked graph in which some vertices represent discrete variables and some represent continuous variables.

... Additional arguments; currently not used.

**adjmat**
An adjacency matrix

**Details**

When `ingraph=TRUE`: If type="decomposable" then `getEdges()` returns those edges e for which the graph with e removed is decomposable.

When `ingraph=FALSE`: Likewise, if type="decomposable" then `getEdges()` returns those edges e for which the graph with e added is decomposable.

The functions `getInEdges()` and `getInEdges()` are just wrappers for calls to `getEdges()`.

The workhorses are `getInEdgesMAT()` and `getOutEdgesMAT()` and these work on adjacency matrices.

Regarding the argument discrete, please see the documentation of `mcsmarked`.

**Value**

A p * 2 matrix with edges.

**Note**

These functions work on undirected graphs. The behaviour is undocumented for directed graphs.

**Author(s)**

Søren Højsgaard, <sorenh@math.aau.dk>

**See Also**

`edgeList`, `nonEdgeList`, `mcsmarked`

**Examples**

```r
gg <- ug(~a:b+d+a:c:d+c:e)
glist <- getClitics(gg)
adjmat <- as.adjMAT(gg)

### On a glist
getEdges(glist)
getEdges(glist, type="decomposable")  # Deleting (a,d) would create a 4-cycle
getEdges(glist, ingraph=FALSE)
getEdges(glist, type="decomposable", ingraph=FALSE)  # Adding (e,b) would create a 4-cycle
```
### On a graphNEL

getEdges(gg)
getEdges(gg, type="decomposable")  
# Deleting (a,d) would create a 4-cycle

getEdges(gg, ingraph=FALSE)
getEdges(gg, type="decomposable", ingraph=FALSE)
# Adding (e,b) would create a 4-cycle

### On an adjacency matrix

getEdges(adjmat)
getEdges(adjmat, type="decomposable")
# Deleting (a,d) would create a 4-cycle

getEdges(adjmat, ingraph=FALSE)
getEdges(adjmat, type="decomposable", ingraph=FALSE)
# Adding (e,b) would create a 4-cycle

## Marked graphs; vertices a,b are discrete; c,d are continuous

UG <- ug(~a:b:c+b:c:d)
disc <- c("a","b")
getEdges(UG)
getEdges(UG, discrete=disc)
## Above: same results; there are 5 edges in the graph

getEdges(UG, type="decomposable")
## Above: 4 edges can be removed and will give a decomposable graph
## (only removing the edge (b,c) would give a non-decomposable model)

getEdges(UG, type="decomposable", discrete=c("a","b"))
## Above: 3 edges can be removed and will give a strongly decomposable
## graph. Removing (b,c) would create a 4--cycle and removing (a,b)
## would create a forbidden path; a path with only continuous vertices
## between two discrete vertices.

---

### ggmfit

*Iterative proportional fitting of graphical Gaussian model*

**Description**

Fit graphical Gaussian model by iterative proportional fitting.

**Usage**

```r
ggmfit( S, n.obs, glist, start=NULL, eps=1e-12, iter=1000, details=0, ...)
ggmfitr(S, n.obs, glist, start=NULL, eps=1e-12, iter=1000, details=0, ...)
```

Arguments

- `S`     Empirical covariance matrix
- `n.obs` Number of observations
- `glist` Generating class for model (a list)
- `start` Initial value for concentration matrix
- `eps`   Convergence criterion
- `iter`  Maximum number of iterations
- `details` Controlling the amount of output.
- `...`  Optional arguments; currently not used

Details

ggmfit is based on a C implementation. ggmfitr is implemented purely in R (and is provided mainly as a benchmark for the C-version).

Value

A list with

- `lrt` Likelihood ratio statistic (-2logL)
- `df` Degrees of freedom
- `logL` log likelihood
- `K` Estimated concentration matrix (inverse covariance matrix)

Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

See Also

- `cmod`, `loglin`

Examples

```r
## Fitting "butterfly model" to mathmark data
## Notice that the output from the two fitting functions is not
## entirely identical.
data(math)
ddd <- cov.wt(math, method="ML")
glist <- list(c("al","st","an"), c("me","ve","al"))
ggmfit (ddd$cov, ddd$n.obs, glist)
ggmfitr(ddd$cov, ddd$n.obs, glist)
```
ghk2phkParms  Conversion between different parametrizations of mixed interaction models.

Description

Functions to convert between canonical parametrization (g,h,K), moment parametrization (p,m,S) and mixed parametrization (p,h,K).

Usage

ghk2phkParms(parms)
ghk2pmsParms(parms)
phk2ghkParms(parms)
phk2pmsParms(parms)
pms2ghkParms(parms)
pms2phkParms(parms)

Arguments

parms Parameters of a mixed interaction model

Value

Parameters of a mixed interaction model.

Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

loglinDim  Return the dimension of a log-linear model

Description

Return the dimension of a log-linear model given by the generating class 'glist'. If the model is decomposable and adjusted dimension can be found.

Usage

loglinGenDim(glist, tableinfo)
loglinDecDim(glist, tableinfo, adjust=TRUE)
**loglinDim**

**Arguments**

- **glist**: Generating class (a list) for a log-linear model. See 'details' below.
- **tableinfo**: Specification of the levels of the variables. See 'details' below.
- **adjust**: Should model dimension be adjusted for sparsity of data (only available for decomposable models)

**Details**

- **glist** can be either a list of vectors with variable names or a list of vectors of variable indices.
- **tableinfo** can be one of three different things:
  1. A contingency table (a table).
  2. A list with the names of the variables and their levels (such as one would get if calling `dimnames` on a table).
  3. A vector with the levels. If `glist` is a list of vectors with variable names, then the entries of the vector `tableinfo` must be named.

If the model is decomposable it `loglinDecDim` is to be preferred over `loglinGenDim` as the former is much faster.

Setting `adjust=TRUE` will force `loglinDecDim` to calculated a dimension which is adjusted for sparsity of data. For this to work, `tableinfo` *MUST* be a table.

**Value**

A numeric.

**Author(s)**

Søren Højsgaard, <sorenh@math.aau.dk>

**See Also**

- `dmod`, `glm`, `loglm`

**Examples**

```r
## glist contains variable names and tableinfo is a named vector:
loglinGenDim(list(c("a","b"),c("b","c")), c(a=4,b=7,c=6))

## glist contains variable names and tableinfo is not named:
loglinGenDim(list(c(1,2),c(2,3)), c(4,7,6))

## For decomposable models:
loglinDecDim(list(c("a","b"),c("b","c")), c(a=4,b=7,c=6), adjust=FALSE)
```
Mixed interaction model.

Description

A mixed interaction model is a model (often with conditional independence restrictions) for a combination of discrete and continuous variables.

Usage

```r
mmod(formula, data, marginal = NULL, fit = TRUE, details = 0)
```

Arguments

- `formula`
- `data`
- `marginal`
- `fit`
- `details`

Value

An object of class `mModel` and the more general class `iModel`.

Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

See Also

dmod cmod

Examples

```r
### FIXME: To be written
```
modify_glist

Modify generating class for a graphical/hierarchical model

Description

Modify generating class for a graphical/hierarchical model by 1) adding edges, 2) deleting edges, 3) adding terms and 4) deleting terms.

Usage

modify_glist(glist, items, details = 0)

Arguments

glist    A list of vectors where each vector is a generator of the model.
items    A list with edges / terms to be added and deleted. See section 'details' below.
details  Control the amount of output (for debugging purposes).

Details

The items is a list with named entries as list(add.edge=, drop.edge=, add.term=, drop.term=)
Not all entries need to be in the list. The corresponding actions are carried out in the order in which they appear in the list.
See section 'examples' below for examples.
Notice that the operations do not in general commute: Adding an edge which is already in a generating class and then removing the edge again does not give the original generating class.

Value

A generating class for the modified model. The elements of the list are character vectors.

Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

See Also

cmod dmod mmod

Examples

glist <- list(c(1,2,3),c(2,3,4))

## Add edges
modify_glist(glist, items=list(add.edge=c(1,4)))
modify_glist(glist, items=list(add.edge=1:4))
## Description

Stepwise model selection in (graphical) interaction models

### Usage

```r
## S3 method for class 'iModel'
stepwise(object,
    criterion = "aic", alpha = NULL, type = "decomposable",
    search = "all", steps = 1000, k = 2,
    direction = "backward", fixinMAT = NULL, fixoutMAT = NULL,
    details = 0, trace = 2, ...)
```

```r
backward(object,
    criterion = "aic", alpha = NULL, type = "decomposable",
    search = "all", steps = 1000, k = 2,
    fixinMAT = NULL, details = 1, trace = 2, ...)
```

```r
forward(object,
    criterion = "aic", alpha = NULL, type = "decomposable",
    search = "all", steps = 1000, k = 2,
    fixoutMAT = NULL, details = 1, trace = 2, ...)
```

### Arguments

- **object**: An iModel model object
- **criterion**: Either "aic" or "test" (for significance test)
- **alpha**: Critical value for deeming an edge to be significant/insignificant. When criterion="aic", alpha defaults to 0; when criterion="test", alpha defaults to 0.05.
stepwise.iModel; backward; forward

Value

An iModel model object.

Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

See Also

cmod dmod mmod testInEdges testOutEdges

Examples

data(reinis)
## The saturated model
m1 <- dmod(~., data=reinis)
m2 <- stepwise(m1)
m2
testadd  

Test addition of edge to graphical model

Description

Performs a test of addition of an edge to a graphical model (an imodel object).

Usage

testadd(object, edge, k=2, details=1, ...)

Arguments

- **object**: A model; an object of class imodel.
- **edge**: An edge; either as a vector or as a right hand sided formula.
- **k**: Penalty parameter used when calculating change in AIC
- **details**: The amount of details to be printed; 0 surpresses all information
- **...**: Further arguments to be passed on to the underlying functions for testing; that is to CItable and CImvn

Details

Let M0 be the model and e=\{u,v\} be an edge and let M1 be the model obtained by adding e to M0. If M1 is decomposable AND e is contained in one clique C only of M1 then the test is carried out in the C-marginal model. In this case, and if the model is a log-linear model then the degrees of freedom is adjusted for sparsity.

Value

A list

Author(s)

Søren Högsgaard, <sorenh@math.au.dk>

See Also

testdelete

Examples

```r
## ## testadd
##
## ## Discrete model
##
data(reinis)
```
testdelete

### A decomposable model
```r
mf <- ~smoke:phys:mental+smoke:systol:mental
object <- dmod(mf, data=reinis)
testadd(object, c("systol","phys"))
```

### A non-decomposable model
```r
mf <- ~smoke:phys:phys:mental+smoke:systol+systol:mental
object <- dmod(mf, data=reinis)
testadd(object, c("phys","systol"))
```

### Continuous model
```r
## A decomposable model
data(math)
mf <- ~me:ve:al+al:an
object <- cmod(mf, data=math)
testadd(object, c("me","an"))
```

### A non-decomposable model
```r
mf <- ~me:ve:al+al:an+an:me
object <- cmod(mf, data=math)
testadd(object, c("me","al"))
```

---

testdelete Test deletion of edge from an interaction model

**Description**
Tests if an edge can be deleted from an interaction model.

**Usage**
```r
testdelete(object, edge, k=2, details=1,...)
```

**Arguments**
- **object** A model; an object of class `imodel`.
- **edge** An edge in the model; either as a right-hand sided formula or as a vector
- **k** Penalty parameter used when calculating change in AIC
- **details** The amount of details to be printed; 0 suppresses all information
- **...** Further arguments to be passed on to the underlying functions for testing; that is to Citable and CImvn
Details

If the model is decomposable and the edge is contained in one clique only then the test is made in the marginal model given by that clique. In that case, if the model is a log-linear model then degrees of freedom are adjusted for sparsity.

Value

A list.

Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

See Also

testadd

Examples

```r
#' # # # # testdelete
#' # # # #
#' # # # Discrete model
#' # #
data(reinis)
#' # A decomposable model
#' #
mf <- ~smoke:phys:mental+smoke:systol:mental
object <- dmod(mf, data=reinis)

testdelete(object,c("phys","mental"))
testdelete(object,c("smoke","mental"))
#testdelete(object,c("systol","phys"))

#' # # # A non-decomposable model
#' #
mf <- ~smoke:phys+phys:mental+smoke:systol+systol:mental
object <- dmod(mf, data=reinis)

testdelete(object,c("phys","mental"))
#testdelete(object,c("systol","phys"))
#testdelete(object,c("smoke","mental"))

#' # # # Continuous model
#' # #
data(math)
#' # A decomposable model
#' #
mf <- ~me:ve:al+me:al:an
object <- cmod(mf, data=math)
```
testInEdges; testOutEdges

Test edges in graphical models with p-value/AIC value

Description

Test edges in graphical models with p-value/AIC value. The models must be iModels.

Usage

testInEdges (object, edgeMAT=NULL, criterion = "aic", k = 2,
alpha = NULL, headlong = FALSE, details = 1, ...)
testOutEdges(object, edgeMAT=NULL, criterion = "aic", k = 2,
alpha = NULL, headlong = FALSE, details = 1, ...)

Arguments

- **object** An iModel model object
- **edgeMAT** A p * 2 matrix with edges
- **criterion** Either "aic" or "test" (for significance test)
- **k** Penalty term when criterion="aic". Only k=2 gives genuine AIC.
- **alpha** Critical value for deeming an edge to be significant/insignificant. When criterion="aic", alpha defaults to 0; when criterion="test", alpha defaults to 0.05.
- **headlong** If TRUE then testing will stop once a model improvement has been found.
- **details** Controls the level of printing on the screen.
- **...** Further arguments to be passed on to testdelete (for testInEdges) and testadd (for testOutEdges).

Value

A matrix.

Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

See Also

ggetEdges, testadd, testdelete
Examples

```r
data(math)
cml <- cm(ve:al:an, data=math)
testInEdges(cml, getEdges(cml$glist))
testOutEdges(cml, getEdges(cml$glist, ingraph=FALSE))
```
Index

+Topic **classes**
dModel-class, 12

+Topic **htest**
ciTtest, 3
ciTtest_df, 4
ciTtest_mvn, 6
ciTtest_ordinal, 7
ciTtest_table, 8
testadd, 24
testdelete, 25
testInEdges; testOutEdges, 27

+Topic **models**
cmod, 10
dmod, 11
effloglin, 13
ggmfit, 16
loglinDim, 18
mmod, 20
stepwise.iModel; backward;
  forward, 22
testadd, 24
testdelete, 25
testInEdges; testOutEdges, 27

+Topic **multivariate**
ggmfit, 16

+Topic **utilities**
CGstats, 2
getEdges, 14
ghk2phkParms, 18
modify_glist, 21
backward (stepwise.iModel; backward;
  forward), 22

CGstats, 2
CGstats_internal (CGstats), 2
chisq.test, 4–6, 9
ciTtest, 3, 5, 6, 8, 9
ciTtest.data.frame, 4, 6, 9
ciTtest.list, 4, 5, 9
ciTtest.table, 4–6
 ciTest, 4, 6, 9
 ciTest_df, 4, 6, 9
 ciTest_mvn, 4, 5, 6, 9
 ciTest_ordinal, 7
 ciTest_table, 4–6, 8, 8
cmod, 10, 12, 17, 20, 21, 23
cModel, 12
CModel-class (dModel-class), 12
coef.mModel (mmod), 20
coefficients.mModel (mmod), 20
cov.wt, 2
dmod, 10, 11, 19–21, 23
dModel-class, 12
edgeList, 15
effloglin, 13
forward (stepwise.iModel; backward;
  forward), 22
getEdges, 14, 27
getInEdges (getEdges), 14
getInEdgesMAT (getEdges), 14
getOutEdges (getEdges), 14
getOutEdgesMAT (getEdges), 14
ggmfit, 10, 16
ggmfitr (ggmfit), 16
ghk2phkParms, 18
ghk2pmsParms (ghk2phkParms), 18
glm, 19
loglin, 14, 17
loglinDecDim (loglinDim), 18
loglinDim, 18
loglinGenDim (loglinDim), 18
loglm, 19
mcsMarked, 15
mmod, 10, 12, 20, 21, 23
mmod_dimension (mmod), 20
mModel, 12
mModel-class (dModel-class), 12
modify_glist, 21
nonEdgeList, 15
oldClass, 12

phk2ghkParms (ghk2phkParms), 18
phk2pmsParms (ghk2phkParms), 18
pms2ghkParms (ghk2phkParms), 18
pms2phkParms (ghk2phkParms), 18
print.CGstats (CGstats), 2
print.citest (ciTest), 3
print.dModel (dmod), 11
print.mModel (mmod), 20
print.testadd (testadd), 24
print.testdelete (testdelete), 25

stepwise.iModel (stepwise.iModel;
   backward; forward), 22
stepwise.iModel; backward; forward, 22
summary.citest (ciTest), 3
summary.mModel (mmod), 20

testadd, 24, 26, 27
testdelete, 24, 25, 27
testEdges (testInEdges; testOutEdges), 27
testInEdges, 23
testInEdges (testInEdges;
   testOutEdges), 27
testInEdges; testOutEdges, 27
testOutEdges, 23
testOutEdges (testInEdges;
   testOutEdges), 27