

Package ‘icbn’

August 3, 2010

Type Package

Title Combine CBNs with isotonic regression of phenotypes.

Version 0.2-13

Date 2010-08-03

Author Patrick Knupfer

Maintainer Patrick Knupfer <pknupfer@ethz.ch>

Description Estimates a CBN where the phenotypes are computed by isotonic regression.

License GPL-2

LazyLoad yes

LazyData yes

Depends R (>= 2.10.0)

Imports isotone, ggm, igraph, e1071

R topics documented:

icbn-package	2
avgPheno	3
checkCompatibility	4
comparePosets	4
computeDistanceMatrix	5
constructPoset	6
convert2ConstraintMatrix	7
convert2EdgeList	8
convertRest2Poset	8
cvIcbn	9
doAnneal	10
doEm	11
estimateEpsilon	12
estimateMu	13
estimateSigma	14
estimateSS	14
estimateThetas	15

exitSet	16
fitCbn	17
generateData	18
generateMus	19
generateThetas	20
getGenotypIdx	21
guessMus	22
guessParams	23
guessThetas	24
hammingDistance	25
hasse	25
hyperCube	26
hyperCubeNames	27
icbnNews	27
map2HyperCube	28
matrixPower	28
mixedRadixGeneration	29
orderIdeals	30
plotIsotonic	31
plotPoset	32
predictPheno	33
printParams	34
probCbn	34
probErr	35
probIso	36
testData	37
testPoset	38
testResponsibilities	38
transitiveClosure	39
tryEdge	40
weightMoves	41
zdv	42

Index	43
--------------	-----------

icbn-package

CBN with isotonic regression on the phenotypes.

Description

Estimates a CBN were the phenotypes are modeled with an isotonic regression.

Details

Package:	icbn
Type:	Package
Version:	0.2-13
Date:	2010-08-03
License:	GPL-2
LazyLoad:	yes
LazyData:	yes

Author(s)

Patrick Knupfer

Maintainer: Patrick Knupfer <pknupfer@ethz.ch>

References

Beerenwinkel N, Eriksson N, Sturmfels B, Bernoulli 13(4), 2007, 893-909

avgPheno

Averages the phenotypes within the genotype lattice.

Description

Takes the average over all phenotypes found for a given genotype.

Usage

```
avgPheno(D)
```

Arguments

D	Data D mapped to the hypercube.
---	---------------------------------

Value

u	The count of phenotypes within the genotype.
ybar	The mean across all phenotypes within the genotype.

Author(s)

Patrick Knupfer

Examples

```
Y <- zdv[,1]
X <- zdv[, -1]
D <- map2HyperCube(X, Y)
aux <- avgPheno(D)
str(aux)
```

`checkCompatibility` *Checks the compatibility of a poset with given data.*

Description

Returns the fraction of the data records which are compatible with the poset.

Usage

```
checkCompatibility(X, P)
```

Arguments

X	The observed data as data frame or matrix.
P	The poset to be checked as a matrix.

Details

Creates an index in the hypercube based on the data. Next the order ideals for the hypercube are determined and the quotient of the `sum(in_model)/sum(all_records)` is returned.

Value

Numeric value containing the fraction of data explained by the poset.

Author(s)

Patrick Knupfer

Examples

```
P <- testPoset
P[1,2] <- 1
checkCompatibility(testData[, -1], hasse(P))
```

`comparePosets` *Compares to posets.*

Description

Return the values of true relations NP, the FPR and FNR.

Usage

```
comparePosets(pTrue, pTest)
```

Arguments

pTrue	The poset considered correct.
pTest	The poset to be compared to the correct one.

Details

The number of relations NP is the number of relations of the transitive closure of the true poset. The FPR is the number of relations in the test poset not in the true poset divided by $(n*(n-1)/2)$. The FNR is the number of relations in the true poset not in the test poset divided by NP. The FNR of an empty poset is set to zero.

Value

NP	True number relations.
FPR	False positive rate.
FNR	False negative rate.

Author(s)

Patrick Knupfer.

Examples

```
P1 <- testPoset
P2 <- matrix(round(runif(16), 0), nc=4)
comparePosets(P1, P2)
```

computeDistanceMatrix

Computes the complete hamming distance matrix between all observations and the hypercube.

Description

Precomputes all distances between observations and the hypercube for further use.

Usage

```
computeDistanceMatrix(X, G, H)
```

Arguments

X	The observed data.
G	Index of the genotypes in the lattice within the hypercube.
H	The hypercube.

Details

In order to speed up things, the distance matrix is calculated only once. The likelihood of the error term is then computed with this matrix as input.

Value

A matrix of dimensions $N \times NH$ with N being the number of observations and NH the number of possible patterns (2^{events}).

Author(s)

Patrick Knupfer

See Also[probErr](#)**Examples**

```
X <- testData[,-1]
H <- hyperCube(ncol(testPoset))
G <- orderIdeals(H, hasse(testPoset))$G
computeDistanceMatrix(X, G, H)
```

constructPoset	<i>construct a poset</i>
----------------	--------------------------

Description

Constructs a poset, given the data, epsilon and the hypercube.

Usage

```
constructPoset(H, D, eps = 1)
```

Arguments

H	Hypercube, all combinations of the p events as a matrix.
D	Data, the observed events mapped to the hypercube.
eps	Epsilon, the maximal allowed violations is the number of observations times epsilon.

Value

poset	The poset
viol	The violations

Note

Ported MATLAB code from Niko Beerenwinkel to R.

Author(s)

Patrick Knupfer

Examples

```

Y <- zdv[,1]
X <- zdv[, -1]
p <- ncol(X)
H <- hyperCube(p)
colnames(H) <- names(X)
D <- map2HyperCube(X, Y)
eps <- 0.1
(P <- constructPoset(H, D, eps)$poset)

```

```
convert2ConstraintMatrix
```

Creates the constraints matrix given the hypercube.

Description

Creates the matrix with the constraints as input to the isotonic regression.

Usage

```
convert2ConstraintMatrix(G, H, coverRelOnly=TRUE)
```

Arguments

G	The index of the order ideals given a poset.
H	The hypercube for the given number of events.
coverRelOnly	Flag if all relations or cover relations should be returned. Defaults to cover relations only.

Value

Returns a matrix with the relations in the form $i < j$.

Author(s)

Patrick Knupfer

Examples

```

H <- hyperCube(nrow(testPoset))
G <- orderIdeals(H, testPoset)$G
convert2ConstraintMatrix(G, H)

```

`convert2EdgeList` *Converts adjacency matrix to an edge list.*

Description

Converts an adjacency matrix to an edge list. Needed to create a graph in igraph. An edge list looks like this 0 1 2 3 for 0 -> 1, 2 -> 3

Usage

```
convert2EdgeList (poset)
```

Arguments

<code>poset</code>	A poset is a quadratic matrix with 0 and 1. A 1 at position (33,99) means 33 comes before 99.
--------------------	---

Value

Returns a vector with the edges. All vertex numbers have 1 subtracted as the numbering in igraph starts with zero.

Author(s)

Patrick Knupfer

Examples

```
## Edges connected as 1->2, 2->3
(P <- testPoset)
(edgeList <- convert2EdgeList(P))
colnames(P) <- rownames(P) <- letters[1:ncol(P)]
P
(edgeList <- convert2EdgeList(P))
```

`convertRest2Poset` *Converts a set of restrictions to a poset.*

Description

Converts a set of restrictions to a poset.

Usage

```
convertRest2Poset (restrMat, p)
```

Arguments

<code>restrMat</code>	An n x 2 matrix with the restrictions.
<code>p</code>	Number of the events in the poset.

Value

Matrix containing the poset.

Author(s)

Patrick Knupfer

Examples

```
restr <- matrix(c(1,2,1,3,2,3,1,5,2,5), nc=2, byrow=TRUE)
pos <- convertRest2Poset(restr, 5)
## plotPoset(pos)
```

cvIcbn	<i>Does cross-validation of noisy, isotonic, conjunctive bayesian network models.</i>
--------	---

Description

Returns MSE and COR of ICBN and linear model.

Usage

```
cvIcbn(fitCbn, X, Y, K, P, epsStop=0.1, maxIter=100)
```

Arguments

fitCbn	Function doing the actual fitting, usually fitCbn.
X	The observed genotypes as binary matrix.
Y	The observed phenotype as numerical vector.
K	The number of cross-validation partitions.
P	The poset under consideration.
epsStop	The EM loop stops if the change of likelihood is below this value.
maxIter	Determines the maximum number of iterations in the isotonic regression.

Details

Returns estimates of the MSE, COR and the respective standard errors for linear and CBN model.

Value

mseLr	MSE of linear model.
mseIcbn	MSE of ICBN model.
corLr	Correlation linear model.
corIcbn	Correlation ICBN model.
seMseLr	Standard error of MSE linear model.
seMseIcbn	Standard error of MSE ICBN model.
seCorLr	Standard error of COR linear model.
seCorIcbn	Standard error of COR ICBN model.
etc	Raw values of mse and cor for testing.

Author(s)

Patrick Knupfer

See Also[fitCbn](#)**Examples**

```
P <- testPoset
X <- testData[, -1]
Y <- testData[, 1]
K <- 3
set.seed(42)
cvIcbn(fitCbn, X, Y, K, P, epsStop=1, maxIter=50)
```

doAnneal

*CBN structure search with simulated annealing.***Description**

The structure searching part encapsulated in a function.

Usage

```
doAnneal(X, Y, outputPath, tInit = 50, tAlpha = 0.03, tFactor = 1, nIter = 10, e
        maxIter = 100, plotFlag = FALSE, saveFlag = FALSE, loggingFlag = FALSE)
```

Arguments

X	The observed genotype as a binary matrix.
Y	The observed phenotype as numeric vector.
outputPath	Output path where the plots and intermediate results are saved.
tInit	Initial temperature in the annealing run.
tAlpha	The cut-off temperature for the shortcut decision.
tFactor	Factor which determines the temperature gradient.
nIter	Number of iterations in the annealing run.
epsGuess	Epsilon for the initial guess of the poset.
epsStop	Epsilon for the EM-step, loop breaking occurs if log likelihood difference below this value.
maxIter	Maximum number of iterations in the isotonic regression.
plotFlag	Save plots to disk?
saveFlag	Save intermediate result to disk?
loggingFlag	Write diagnostic messages to the console.

Details

Returns the parameter set including poset of the configuration in the annealing run which showed the highest likelihood.

Value

thetas	Optimal thetas found.
epsilon	Optimal epsilon found.
mus	Optimal mus found.
sigma2	Optimal sigma2 found.
P	Optimal poset found.
G	Genotype lattice indices in the hypercube.
H	Hypercube.
respMat	Responsibility matrix.
distMat	Distance matrix.
lObs	Observed log likelihood.

Author(s)

Patrick Knupfer.

Examples

```
Y <- testData[,1]
X <- testData[,-1]
set.seed(42)
doAnneal(X, Y, "myPathName", tInit = 100, tAlpha = 0.03, tFactor = 1, nIter = 10, epsGuess = 0.1,
          maxIter = 50, plotFlag = FALSE, saveFlag = FALSE, loggingFlag = FALSE)
```

doEm

EM algorithm

Description

Finds the maximum likelihood parameters theta, mu, sigma and epsilon and the responsibilities.

Usage

```
doEm(X, Y, params, eps = 0.1, maxIter=100, logging=TRUE)
```

Arguments

X	The observed genotypes.
Y	The observed phenotypes, continuous numerical values.
params	A list containing the parameters theta, epsilon, mu, theta, P, G and H.
eps	Minimal difference in log likelihood before convergence.
maxIter	Maximum number of iterations in the isotonic regression estimation.
logging	Flag to turn on messages to the console.

Details

Starting with guessed/estimated parameters performs an EM algorithm and returns the improved parameter estimates, the likelihood and the responsibilities.

Value

A list with the following components:

`params` A list containing the parameters after the last iteration.

Author(s)

Patrick Knupfer

Examples

```
P <- testPoset
Y <- testData[,1]
X <- testData[,-1]
params <- guessParams(X, Y, P, thetas=NULL, epsilon=NULL, mus=NULL, sigma2=NULL)
set.seed(42)
params <- doEm(X, Y, params, eps = 0.5, maxIter=50, logging=TRUE)
```

<code>estimateEpsilon</code>	<i>Estimates the error coefficient.</i>
------------------------------	---

Description

Estimation of the error coefficient given the responsibility matrix. The row names have to encode the observed genotype, the column names the genotypes in the lattice.

Usage

```
estimateEpsilon(X, G, H, respMat)
```

Arguments

<code>X</code>	The observed data.
<code>G</code>	The index of genotypes in the lattice.
<code>H</code>	The full hypercube.
<code>respMat</code>	$N * n$ matrix with the responsibilities

Value

The estimate of epsilon, the error coefficient.

Author(s)

Patrick Knufer

Examples

```

set.seed(42)
P <- testPoset
Y <- testData[,1]
X <- testData[,-1]
H <- hyperCube(nrow(P))
G <- orderIdeals(H, P)$G
params <- guessParams(X, Y, P, thetas=NULL, epsilon=NULL, mus=NULL, sigma2=NULL)
respMat <- doEm(X, Y, params, eps = 0.5, maxIter=50, logging=FALSE)$respMat
(eps <- estimateEpsilon(X, G, H, respMat))

```

estimateMu

*Estimates the parameters mu using isotonic regression***Description**

Uses the package isotone.

Usage

```
estimateMu(Y, G, H, respMat, maxIter=100, logging=TRUE)
```

Arguments

Y	Vector with the measured numeric phenotypes.
G	Genotype lattice indicating membership of g in G.
H	The hypercube of possible genotypes.
respMat	The responsibility matrix calculated in the E-step.
maxIter	Maximum number of iterations in the isotonic regression estimation.
logging	Flag to turn on messages to the console.

Details

The responsibility matrix has dimensions N times n with N being the number of observations and n being the number of genotypes in the genotype lattice. The values of y are transformed to $\sum_i(\gamma_{ig} \mu_g) / \sum_i(\gamma_{ig})$, the values of Ng are replaced by ug.

Value

A numeric vector with the estimated parameters mu. Dimension corresponds to the number of genotypes in the lattice.

Author(s)

Patrick Knupfer

Examples

```

P <- testPoset
Y <- testData[,1]
H <- hyperCube(ncol(P))
G <- orderIdeals(H, P)$G
estimateMu(Y, G, H, testResponsibilities)

```

estimateSigma	<i>Estimates the variance of the isotonic regression.</i>
---------------	---

Description

Estimation of the variance from the isotonic regression.

Usage

```
estimateSigma(Y, mus, respMat)
```

Arguments

Y	The measured numeric values of the phenotype.
mus	The estimated mean values of the phenotypes by genotype in the lattice.
respMat	The N times n matrix with the responsibilities calculated in the E step.

Details

The number of values in the vector mus depends on the genotypes in the lattice.

Value

A single number containing $(\sigma_{\text{hat}})^2$.

Author(s)

Patrick Knupfer

Examples

```
P <- testPoset
Y <- testData[,1]
H <- hyperCube(ncol(P))
G <- orderIdeals(H, P)$G
mus <- estimateMu(Y, G, H, testResponsibilities)
estimateSigma(Y, mus, testResponsibilities)
```

estimateSS	<i>Get the sum of squares error of the estimated phenotypes.</i>
------------	--

Description

Determines the SS of a linear, constant and noisy iCBN model.

Usage

```
estimateSS(X, Y, params)
```

Arguments

X	The observed genotypes.
Y	The observed phenotypes.
params	The parameters fitted during the EM algorithm or the last output of the simulated annealing.

Details

The iCBN should do similar or better than the linear model, i.e. the SS must be of the same order.

Value

ssConst	SS of the constant model.
ssLr	SS of the linear regression.
ssIcbn	SS of the noisy isotonic conjunctive bayesian network.
lObs	The observed log likelihood, i.e. $\log(L(X,Y))$.
y	Vector of observed phenotypes.
yhatConst	Mean value of the observed phenotypes.
yhatLr	Vector with predicted phenotypes using linear regression.
yhatIcbn	Vector with predicted phenotypes using noisy ICBN.

Author(s)

Patrick Knupfer

Examples

```
P <- testPoset
Y <- testData[,1]
X <- testData[,-1]
params <- guessParams(X, Y, P, thetas=NULL, epsilon=NULL, mus=NULL, sigma2=NULL)
set.seed(42)
params <- doEm(X, Y, params, eps = 0.5, maxIter=50, logging=TRUE)
estimateSS(X, Y, params)
```

estimateThetas	<i>Estimates the conditional probabilities of a CBN, given the cover relations and the counts per genotype.</i>
----------------	---

Description

Estimates the conditional probabilities in a CBN, given the hasse poset and the genotype counts.

Usage

```
estimateThetas(P, G, H, respMat)
```

Arguments

P	Quadratic matrix containing the cover relations.
G	Index vector of reachable genotypes in the hypercube.
H	Hypercube with all possible genotypes.
respMat	The responsibility matrix used to get the expected counts.

Value

Returns a vector containing the thetas.

Author(s)

Patrick Knupfer

Examples

```
P <- testPoset
H <- hyperCube(ncol(P))
G <- orderIdeals(H, P)$G
estimateThetas(P, G, H, testResponsibilities)
```

exitSet

Extracts the exit sets for a given genotype.

Description

Exit = exitSet(G, g) returns the exit set of g, i.e., $\text{Exit} = \min(g^c)$ is the set of minimal elements of the complement of g in the event set w.r.t. to a partial order P with lattice of order ideals $G = J(P)$.

Usage

```
exitSet(G, g)
```

Arguments

G	The order ideals $G = J(P)$ with P being the partial order.
g	The genotype considered.

Value

The exit sets as a list.

Note

Ported MATLAB code from Niko Beerenwinkel to R.

Author(s)

Patrick Knupfer

Examples

```
P <- testPoset
H <- hyperCube(ncol(P))
G <- orderIdeals(H, P)$G
g <- c(0, 0, 0)
exitSet(G, g)
```

fitCbn

*Helper function for cross-validation of CBN.***Description**

Returns the MSE of a linear and CBN fit as well as the correlations.

Usage

```
fitCbn(idx, X, Y, P, epsStop=0.1, maxIter=100)
```

Arguments

<code>idx</code>	Row indices to use as validation, the rest is used for training.
<code>X</code>	The observed genotypes as binary matrix.
<code>Y</code>	The observed phenotypes as numerical vector.
<code>P</code>	The poset under consideration.
<code>epsStop</code>	EM loop stops if difference in likelihood is below this value.
<code>maxIter</code>	Determines the maximum number of iterations in the isotonic regression.

Details

Splits the data into training and validation. Initially guesses CBN parameters and fits them in EM algorithm. Determines the beta values of a linear regression. Parameter estimates are done on training portion, predictions on validation portion. Calls the function `predictPheno` to obtain predictions and calculate MSE and COR for the CBN and LR model.

Value

<code>mseLr</code>	MSE of linear regression.
<code>mseIcbn</code>	MSE of CBN model.
<code>corLr</code>	Correlation of linear model prediction with observed data.
<code>corIcbn</code>	Correlation of CBN model prediction with observed data.

Author(s)

Patrick Knupfer

See Also

[predictPheno](#)

Examples

```
set.seed(42)
P <- testPoset
Y <- testData[,1]
X <- testData[,-1]
idx <- sample(1:nrow(X), nrow(X)/2)
fitCbn(idx, X, Y, P, epsStop=0.1, maxIter=100)
```

generateData	<i>Data generator for testing.</i>
--------------	------------------------------------

Description

Generates data through a noisy isotonic CBN model. There are 3 components: The CBN determining $P(Z)$, an error models determining $P(X|Z)$ and an isotonic or linear regression determining $P(Y|Z)$.

Usage

```
generateData(P, thetas, epsilon, mus, sigma2, N, betas=NULL, isoReg=TRUE)
```

Arguments

P	The poset determining the CBN structure.
thetas	A vector containing the parameters of the CBN, one value of theta per event in the model.
epsilon	A parameter determining the error probabilities.
mus	A vector containing the average phenotype values associated with a genotype in the lattice. The number of parameters is given by the number of genotypes in the lattice, i.e. $\text{sum}(G)$
sigma2	The variance of phenotypes belonging to the same lattice. One value for all genotypes.
N	The number of observations to generate.
betas	The parameters for the linear regression. Length = Number of events + 1 for the intercept.
isoReg	Flag to define which regression model should be used, isotonic or linear. Defaults to isotonic (=TRUE).

Details

Bases on the values of thetas, each event $E_1..E_n$ is drawn from $\text{Bernoulli}(\text{theta}_1..\text{theta}_n)$. Using the poset the event occurrence is set to $E_i * E(\text{parents}(E_i))$ to ensure only events are generated where all parents have occurred. Next a flip event E_{err} is drawn from $\text{Bernoulli}(\text{epsilon})$ and the events are inverted accordingly. Last for each genotype a phenotype value is drawn from a normal distribution based on μ value of the corresponding hidden genotype and the value of sigma2 . In case of linear regression, y is drawn from $N([1,g] * \text{betas}, \text{sigma2})$.

Value

A data frame with the first column being the numeric phenotype value. The next N columns represent the observed occurrence of the N events given in the poset. The next N columns represent the true occurrence of the N events given in the poset.

Author(s)

Patrick Knupfer

Examples

```
## Specifiy poset
P <- matrix(c(0,1,0,0,
0,0,0,0,
0,1,0,1,
0,0,0,0), nc=4, byrow=TRUE)
P <- hasse(transitiveClosure(P))
colnames(P) <- rownames(P) <- paste("e", 1:ncol(P), sep="")
## Make sure it's correct
# plotPoset(P, vertexLabels=colnames(P))

## Extract the hypercube and order ideals

H <- hyperCube(ncol(P))
rownames(H) <- hyperCubeNames(ncol(P))
G <- orderIdeals(H, hasse(P))$G
cat("Number of order ideals: ", sum(G))

## Set parameters for data generation

thetas <- c(0.4, 0.6, 0.7, 0.8)
mus <- c(0.0,0.1,0.2,0.1,0.2,0.3,0.3,0.4)
names(mus) <- hyperCubeNames(nrow(P))[as.logical(orderIdeals(hyperCube(nrow(P)), P)$G)]
epsilon <- 0.2
sigma2 <- 0.1
N <- 200

## Generate the data

cbnData <- generateData(P, thetas, epsilon, mus, sigma2, N)
str(cbnData)
```

generateMus

Generates random mu values for the isotonic regression.

Description

The generated random values are assigned to the genotypes in the lattice by increasing number of mutations.

Usage

```
generateMus(P, minimum, maximum, graded=TRUE, scaled=TRUE)
```

Arguments

P	The poset as a matrix.
minimum	Minimum value of the mus to be generated.
maximum	Maximal value of the mus to be generated.
graded	Should the mus be the same for the same number of mutations.
scaled	Should the generated mu values be scaled.

Details

A number of random uniform values is generated corresponding to the number of genotypes in the lattice and sorted. Then the values are assigned to the genotypes sorted by the number of mutations. If option graded is true, the same random value will be assigned to all genotypes with the same number of mutations. If option scaled is true, the wild type and fully mutated genotypes are set to the -1 and 3 respectively.

Value

A numeric vector containing the randomly generated mu values.

Author(s)

Patrick Knupfer

Examples

```
P <- testPoset
generateMus(P, -2, 2)
```

generateThetas	<i>Generates uniformly distributed values of theta.</i>
----------------	---

Description

Generates the required number of thetas in the specified range.

Usage

```
generateThetas(p, minimum = 0.5, maximum = 0.9)
```

Arguments

p	Number of theta values to be generated.
minimum	Minimum value of theta.
maximum	Maximum value of theta.

Details

Simply draws uniform random values from the specified interval. Can be modified to draw thetas in a more elaborate way.

Value

Numeric vector containing the thetas.

Author(s)

Patrick Knupfer

Examples

```
generateThetas(5)
```

getGenotypeIdx	<i>Given a genotype extracts the index in a lexically ordered hypercube.</i>
----------------	--

Description

Returns the index of a genotype in the hypercube.

Usage

```
getGenotypeIdx(Z)
```

Arguments

Z	Matrix containing the genotypes.
---	----------------------------------

Value

Returns a numerical vector containing the indices.

Author(s)

Patrick Knupfer

Examples

```
Z <- matrix(c(0,0,0,0,1,0,1,1,0,1,1,1), nc=3, byrow=TRUE)
hyperCubeNames(ncol(Z))
getGenotypeIdx(Z)
hyperCubeNames(ncol(Z))[getGenotypeIdx(Z)]
```

guessMus

Initial guess of the parameters of the isotonic regression.

Description

In order to speed up convergence, the mu values from the previous step are reused. As the poset changes, the old mus are mapped to the new ones, for non-existing genotypes the mus are set in order not to violate the isotonicity constraints.

Usage

```
guessMus(musOld, G, H)
```

Arguments

musOld	Named numeric vector with the previous parameters according to the OLD poset.
G	The index of genotypes in the lattice with respect to the hypercube according to the NEW poset.
H	The hypercube.

Details

For new genotypes in the lattice, the phenotype values up- or downstream are searched and the corresponding values are taken.

Value

Named numeric vector containing the new parameters. The length is usually different from the length of the old parameters.

Author(s)

Patrick Knupfer

Examples

```
pOld <- testPoset
(musOld <- generateMus(pOld, -1, 3))
pNew <- matrix(c(0,0,1,0,0,0,1,1,0,0,0,0,0,0,0), nc=4, byrow=TRUE)
p <- ncol(pNew)
hNew <- hyperCube(ncol(pNew))
gNew <- orderIdeals(hNew, pNew)$G
guessMus(musOld, gNew, hNew)
```

guessParams

*Sets initial guesses for the parameters to be estimated.***Description**

Sets initial values for the parameters to be estimated.

Usage

```
guessParams(X, Y, P, thetas=NULL, epsilon=NULL, mus=NULL, sigma2=NULL)
```

Arguments

X	The observed genotypes.
Y	The observed pheonotypes.
P	The current poset.
thetas	The values of the thetas to be assumed.
epsilon	The error rate to be assumed.
mus	The phenotype values in the isotonic regression setup.
sigma2	The variance term of the isotonic regression.

Details

Simply sets the supplied values. If a value is NULL, defaults are used. The vector mu is modelled as a function of the number of mutations. Sigma2 is set to the overall variance of Y.

Value

A list with the initial parameter values.

thetas	Initial thetas.
epsilon	Initial epsilon.
mus	Initial mus.
sigma2	Initial sigma2.
P	The poset P.
G	The index of the order ideals in the hypercube.
H	The hypercube.
respMat	The responsibility matrix.
distMat	The distance matrix.
lObs	The observed log likelihood.

Author(s)

Patrick Knupfer

Examples

```
P <- testPoset
Y <- testData[,1]
X <- testData[,-1]
(params <- guessParams(X, Y, P, thetas=NULL, epsilon=NULL, mus=NULL, sigma2=NULL))
str(params)
```

guessThetas

Guessing the theta parameters without responsibilities.

Description

Estimate the theta parameters of a CBN without responsibilities.

Usage

```
guessThetas(P, G, H, X, Y)
```

Arguments

P	The poset.
G	Index of the genotypes in the lattice relative to the hypercube.
H	The hypercube.
X	The observed genotypes.
Y	The observed phenotypes.

Details

Initial estimate of the thetas in order to obtain good starting values for the EM algorithm.

Value

A numeric vector with the theta values.

Author(s)

Patrick Knupfer

Examples

```
P <- testPoset
Y <- testData[,1]
X <- testData[,-1]
H <- hyperCube(ncol(P))
G <- orderIdeals(H, P)$G
guessThetas(P, G, H, X, Y)
```

hammingDistance	<i>Calculates the Hamming distance between two binary vectors.</i>
-----------------	--

Description

The number of differences in the two input vectors. Character vectors will be converted to numeric.

Usage

```
hammingDistance(x, y)
```

Arguments

x	Vector containing ones and zeroes.
y	Vector containing ones and zeroes.

Details

Simple count of the differences.

Value

Returns the number of differences.

Author(s)

Patrick Knupfer

Examples

```
x <- c(1, 1, 1, 0, 0, 0)
y <- c(1, 0, 1, 0, 0, 1)
hammingDistance(x, y)
```

hasse	<i>Cover relations of a poset.</i>
-------	------------------------------------

Description

Returns the cover relations of a poset.

Usage

```
hasse(poset)
```

Arguments

poset	The poset in form of a matrix.
-------	--------------------------------

Value

The cover relations of the poset.

Note

Ported MATLAB code from Niko Beerenwinkel to R.

Author(s)

Patrick Knupfer

Examples

```
(pTrans <- matrix(c(0,1,1,1,0,0,1,1,0,0,0,1,0,0,0,0), nc=4, byrow=TRUE))
vertexLabels <- paste("Event",1:ncol(pTrans), sep="_")
(pHasse <- hasse(pTrans))
#par(mfrow=c(1,2))
#plotPoset(pTrans, vertexLabels=vertexLabels, hasse=TRUE)
#plotPoset(pTrans, vertexLabels=vertexLabels, hasse=FALSE)
```

hyperCube

Creates a hypercube given the number of events n.

Description

hyperCube(n) returns the 2^n elements of the n-dimensional hypercube in lexicographic order.

The index of a binary vector x of length n in the hypercube H is

$1 + \sum([2^{(n-1)} 2^{(n-2)} \dots 2^0] .* x)$ such that $H(1 + \sum(\text{pow2}(x, [n-1 \ n-2 \dots 0])), :) == x$

Usage

```
hyperCube(n)
```

Arguments

n The number of events.

Details

The hypercube is returned in lexicographic order as a matrix. It is assumed that the events are binary.

Value

H The matrix H representing the hypercube with n events.

Note

Ported MATLAB code from Niko Beerenwinkel to R.

Author(s)

Patrick Knupfer

Examples

```
hyperCube(3)
```

hyperCubeNames	<i>Creates the names of the hypercube elements in lexicographic order.</i>
----------------	--

Description

Creates a vector of mode character enumerating the elements of the hypercube.

Usage

```
hyperCubeNames(n)
```

Arguments

n	The number of events n.
---	-------------------------

Value

Vector with the hypercube elements represented as character strings.

Author(s)

Patrick Knupfer

Examples

```
(aux <- hyperCubeNames(3))  
str(aux)
```

icbnNews	<i>Show the NEWS file</i>
----------	---------------------------

Description

Show the NEWS file of the icbn package.

Usage

```
icbnNews()
```

Value

None.

map2HyperCube	<i>Maps the observed phenotypes to the hypercube.</i>
---------------	---

Description

Maps observations to data structure indexed by hypercube elements

Usage

```
map2HyperCube(X, Y)
```

Arguments

X	Matrix with the observed genotypes.
Y	Vector of observed phenotypes.

Value

D, the data in a data structre indexed by the hypercube elements.

Note

Ported MATLAB code from Niko Beerenwinkel to R.

Author(s)

Patrick Knupfer

Examples

```
Y <- zdv[,1]
X <- zdv[,-1]
(D <- map2HyperCube(X,Y))
```

matrixPower	<i>Fast algorithm to calculate the power of a matrix</i>
-------------	--

Description

Calculates the power of a matrix. Code taken from Package Biodem.

Usage

```
matrixPower(X, n)
```

Arguments

X	A square matrix
n	The power the matrix should be raised to.

Details

This function calculates (efficiently!) the n-th power of a matrix.

Value

Returns a squared matrix.

Note

Included in this package to avoid dependency on Biomed package.

Author(s)

Vincente Canto Cassola and Martin Maechler

Examples

```
M <- matrix(c(1,2,3,4), nc=2)
(M %*% M %*% M %*% M)
(matrixPower(M, 4))
```

`mixedRadixGeneration`

Generates n-tuples in lexicographic order.

Description

Generate all n-tuples (a₁, ..., a_n) with 0 ≤ a_j < m_j for all j = 1, ..., d in lexicographic order.

Usage

```
mixedRadixGeneration(d, m)
```

Arguments

d	Number of elements in a tuple.
m	Cardinality of an event as a vector, use [2 2 2] for binary events.

Value

Matrix representing the hypercube.

Note

Ported MATLAB code from Niko Beerenwinkel to R.

Author(s)

Patrick Knupfer

References

Algorithm M, Knuth 7.2.1.1, page 2

Examples

```
mixedRadixGeneration(3, c(2, 2, 2))
```

orderIdeals	<i>Order ideals of a poset.</i>
-------------	---------------------------------

Description

Extracts the order ideals of a poset given the hypercube.

Usage

```
orderIdeals(H, poset)
```

Arguments

H	The hypercube of all possible events, given as a matrix of 1 and 0 with p columns and 2^p rows.
poset	The poset.

Details

Returns the order ideals.

Value

G	The list of order ideals.
Gadj	The adjacency matrix of order ideals.

Note

Ported MATLAB code from Niko Beerenwinkel to R.

Author(s)

Patrick Knupfer

Examples

```
Y <- zdv[,1]
X <- zdv[,-1]
p <- ncol(X)
H <- hyperCube(p)
D <- map2HyperCube(X, Y)
eps <- 0.1
P <- constructPoset(H, D, eps)$poset
aux <- orderIdeals(H, P[[1]])
str(aux)
```

plotIsotonic	<i>Diagnostic plot of the genotype lattice with the fitted values of the isotonic regression.</i>
--------------	---

Description

Plot the genotype lattice, vertex colors are controlled by quantiles of the phenotype values predicted by isotonic regression. Vertex labels are given by concatenating the binary representation of the pattern with the phenotype values.

Usage

```
plotIsotonic(mus, G, H, eventNames, ug=NULL, layout=NULL, colorDataQuant=FALSE,
```

Arguments

mus	The values fitted by isotonic regression. Should have a names attribute with the binary representation of the pattern.
G	The index values of the genotype lattice in the hypercube.
H	The hypercube.
eventNames	Vector of type character containing the event names for use in the plot title.
ug	The count of observations for that genotype in the lattice. Used for annotating the vertices.
layout	A string specifying the layout. Possible values are: "kamada", "lgl", "reingold", NULL is interpreted as "circle".
colorDataQuant	If set, the vertex colors are scaled to quantiles based on the data, else they are scaled to the range -1 to 3.
noMus	If set to true, the values of ug and the mu are not printed in the vertices of the graph and only the pattern is printed in the vertex.
...	Additional paramters passed to plotPoset.

Value

A diagnostic plot of the icbn lattice.

Author(s)

Patrick Knupfer

Examples

```
P <- testPoset
mus <- generateMus(P, -1, 3)
H <- hyperCube(ncol(P))
G <- orderIdeals(H, P)$G
#plotIsotonic(mus, G, H, paste("Event", 1:ncol(P), sep=""), layout="reingold")
```

plotPoset	<i>Plots the hasse diagram of a poset.</i>
-----------	--

Description

Takes a poset as matrix and draws the hasse diagram.

Usage

```
plotPoset(poset, vertexLabels = NULL, hasse = TRUE, layout=NULL, ...)
```

Arguments

poset	The poset as a squared matrix.
vertexLabels	Optional labels for the vertices. Check the order of the labels.
hasse	Switch to draw cover relations only. Defaults to hasse=TRUE.
layout	A string specifying the layout. Possible values are: "kamada", "lgl", "reingold", NULL is interpreted as "circle".
...	Remaining parameters passed to the plot.igraph function.

Value

No return value.

Note

Use posets without colnames and rownames as the edge list returned cannot be read by graph.

Author(s)

Patrick Knupfer

Examples

```
(pTrans <- matrix(c(0,1,1,1,0,0,1,1,0,0,0,1,0,0,0,0), nc=4, byrow=TRUE))
vertexLabels <- paste("Event",1:ncol(pTrans), sep="_")
(pHasse <- hasse(pTrans))
#par(mfrow=c(1,2))
#plotPoset(pTrans, vertexLabels=vertexLabels, hasse=TRUE)
#plotPoset(pTrans, vertexLabels=vertexLabels, hasse=FALSE)
```

predictPheno	<i>Predicts new data based on CBN model, linear regression and mean value.</i>
--------------	--

Description

Used to assess performance of the CBN model.

Usage

```
predictPheno(xNew, yNew, parIcbn, parLr)
```

Arguments

xNew	New observed genotypes.
yNew	New observed phenotypes.
parIcbn	The CBN parameters based on the training data.
parLr	Linear model parameters based on training data.

Value

A list containing various predictions.

Author(s)

Patrick Knupfer

Examples

```
X <- zdv[, -1]
Y <- zdv[, 1]
P <- hasse(constructPoset(hyperCube(ncol(X))), map2HyperCube(X, Y), eps=0.05)$poset[[1]])
parIcbn <- guessParams(X, Y, P)
parIcbn <- doEm(X, Y, parIcbn, eps=1, maxIter=50)
printParams(parIcbn, "Fitted parameters")
## Predict mus for ICBN and LR
lrMod <- lm(Y~., data=X)
parLr <- lrMod$coef
prediction <- predictPheno(X, Y, parIcbn, parLr)
str(prediction)
```

printParams	<i>Prints out the parameter values.</i>
-------------	---

Description

Prints out the parameter values.

Usage

```
printParams(params, message=NULL, extended=FALSE)
```

Arguments

params	The paramter object from the EM algorithm.
message	Optional message which is printed to the console.
extended	Flag to print additional stuff to the console. Defaults to FALSE.

Details

Prints out the parameter values.

Value

Entries in the console.

Author(s)

Patrick Knupfer

Examples

```
P <- testPoset
Y <- testData[,1]
X <- testData[,-1]
params <- guessParams(X, Y, P, thetas=NULL, epsilon=NULL, mus=NULL, sigma2=NULL)
printParams(params, message="Message", extended=TRUE)
```

probCbn	<i>Calculates the likelihood of a CBN.</i>
---------	--

Description

Returns $\text{Prob}(Z_i=g \mid \text{thetas})$ using the formula $\text{Prod_e_in_g} \theta_{g,e} * \text{Prod_e_in_Succ}(g) (1-\theta_{g,e})$
 Referred to as P_cbn

Usage

```
probCbn(G, H, thetas)
```

Arguments

G	A vector of 1 and 0 specifying which genotypes of H are in the genotype lattice.
H	The hypercube as a matrix containing all possible 2^n genotypes.
thetas	The conditional probability vector. The length corresponds to the number of events in the event poset.

Details

Check Sturmfels et. al. or Beerenwinkel et. al. for details on CBN.

Value

A numeric vector containing $P(Z_i=g|thetas)$ for all g in G.

Author(s)

Patrick Knupfer

Examples

```
P <- hasse(testPoset)
H <- hyperCube(ncol(P))
G <- orderIdeals(H, P)$G
thetas <- c(0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7)
probCbn(G, H, thetas)
```

probErr	<i>Fast version to calculate the probabilities of the error term. $P(x_i z_i=g, \epsilon, p) = \text{choose}(p, d) * \epsilon^d * (1-\epsilon)^{(p-d)}$ with d being the hamming distance between x_i and z_i. These distances can be calculated only once and are stored in distanceMatrix.</i>
---------	--

Description

Uses a precomputed distance matrix on the whole hypercube to save time.

Usage

```
probErr(epsilon, distanceMatrix, p)
```

Arguments

epsilon	The numeric value of epsilon.
distanceMatrix	The distance matrix of all observed genotypes against the genotypes in the lattice genotypes.
p	The number of events.

Details

Instead of computing the hamming distances over and over again, this is done only once and the results stored in a distance matrix which is then used as input to this function which evaluates the error function in a vectorized way.

Value

The matrix elements $P(xilzi=g, \text{epsilon}, p)$ also referred to as P_err . Dimension is $N \times NG$ where N is the number of observations and NG the number of genotypes in the lattice.

Author(s)

Patrick Knupfer

See Also

[computeDistanceMatrix](#)

Examples

```
P <- hasse(testPoset)
X <- testData[, -1]
H <- hyperCube(ncol(P))
G <- orderIdeals(H, P)$G
D <- computeDistanceMatrix(X, G, H)
probErr(0.1, D, ncol(P))
```

probIso	<i>Calculates the probability $P(yilzi=g, \text{mus}, \text{sigma}2)$ where $P(yilzi=g)$ is distributed as $\text{Normal}(\text{mu}_g, \text{sigma}2)$</i>
---------	---

Description

Calculates the likelihood part of the isotonic regression.

Usage

```
probIso(Y, mus, sigma2)
```

Arguments

Y	The observed phenotypes Y.
mus	Parameters of the isotonic regression normal distribution.
sigma2	Parameter of the isotonic regression normal distribution.

Value

A matrix containing the probabilities $P(yilzi=g, \text{mus}, \text{sigma}2)$

Author(s)

Patrick Knupfer

Examples

```

Y <- zdv[,1]
X <- zdv[,-1]
P <- matrix(c(0,0,1,0,1,1,0,
0,0,1,0,0,0,1,
0,0,0,0,0,0,0,
0,0,1,0,0,0,1,
0,0,1,0,0,0,0,
0,0,1,0,1,0,0,
0,0,1,0,0,0,0), nc=7, nr=7, byrow=TRUE)
P <- hasse(P)
names <- c('41L','67N','69D','70R','210W','215Y','219Q')
# plotPoset(P, vertexLabels=names, hasse=TRUE)
H <- hyperCube(ncol(P))
G <- orderIdeals(H, P)$G
mus <- 1:sum(G)
names(mus) <- hyperCubeNames(ncol(P))[as.logical(G)]
sigma2 <- 0.2
prob <- probIso(Y, mus, sigma2)
str(prob)

```

testData

*Test data for cross-checking against CBN paper.***Description**

4 events, genotype lattice with 7 nodes, 37 observations.

Usage

```
data(testData)
```

Format

A data frame with 37 observations on the following 5 variables.

y Numeric phenotype
e1 Binary event 1
e2 Binary event 2
e3 Binary event 3
e4 Binary event 4

Details

For testing purposes.

Source

Made up data giving the results in the CBN paper.

Examples

```
str(testData)
```

testPoset

Poset for the test data set from the CBN paper.

Description

4 events, restrictions $1 < 3$, $1 < 4$, $2 < 3$, $2 < 4$

Usage

```
data(testPoset)
```

Format

The format is: num [1:4, 1:4] 0 0 0 0 0 0 0 1 1 ...

Examples

```
str(testPoset)
```

testResponsibilities

Perfect responsibilities for the test data set from the CBN paper.

Description

Responsibility is one for the correct genotype and zero otherwise.

Usage

```
data(testResponsibilities)
```

Format

The format is: num [1:37, 1:7] 1 1 1 1 1 1 1 0 0 0 ...

Details

37 observations, 7 genotypes in the lattice.

Examples

```
str(testResponsibilities)
```

transitiveClosure	<i>Transitive closure of a set of relations.</i>
-------------------	--

Description

Returns the transitive closure of a set of relations, i.e., the minimal poset containing the relations

Usage

```
transitiveClosure(poset)
```

Arguments

poset	The poset as a matrix.
-------	------------------------

Details

Returns the transitive closure of a set of relations.

Value

poset	The transitive closure of the poset.
-------	--------------------------------------

Note

Ported MATLAB code from Niko Beerenwinkel to R.

Author(s)

Patrick Knupfer

Examples

```
(pHasse <- matrix(c(0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0), nc=4, byrow=TRUE))
vertexLabels <- paste("Event",1:ncol(pHasse), sep="_")
(pTrans <- transitiveClosure(pHasse))
#par(mfrow=c(1,2))
#plotPoset(pTrans, vertexLabels=vertexLabels, hasse=TRUE)
#plotPoset(pTrans, vertexLabels=vertexLabels, hasse=FALSE)
```

tryEdge	<i>Tries modifying an edge.</i>
---------	---------------------------------

Description

Either adds or deletes an edge.

Usage

```
tryEdge(X, Y, params, T, tAlpha=0.05, logLikNext, weightsNext, parsNext, eps=1,
```

Arguments

X	The observed genotype X.
Y	The observed phenotype Y.
params	The initial parameter, containing theta, epsilon, mu, sigma2, P, G and H.
T	The temperature used for the annealing process.
tAlpha	The temperature used for the preselection based on compatibilities.
logLikNext	Matrix with precalculated log likelihoods.
weightsNext	Matrix of the fractions of explained genotypes.
parsNext	List of precalculated parameters.
eps	Maximal log likelihood difference at last step.
maxIter	Maximum number of iterations in the isotonic regression.
logging	Flag to turn on messages to the console.

Details

Tries adding or deleting an edge.

Value

A list with the following components:

params	A list containing the new/old parameters, depending on the acceptance/rejection of the proposal.
logLikNext	Matrix with the precalculated log likelihoods, is filled with zeros if the last change was accepted and the poset changed.
parsNext	List with precalculated parameter sets, is an empty list if the last change was accepted and the poset changed.

Author(s)

Patrick Knupfer

Examples

```

P <- testPoset
X <- testData[, -1]
Y <- testData[, 1]
set.seed(42)
params <- guessParams(X, Y, P, thetas=NULL, epsilon=NULL, mus=NULL, sigma2=NULL)
logLikNext <- matrix(0, nc=ncol(P), nr=ncol(P))
weightsNext <- NULL
parsNext <- list()
edgeResult <- tryEdge(X, Y, params, 100, tAlpha=0.05, logLikNext, weightsNext, parsNext,
eps=1, maxIter=50, logging=TRUE)
str(edgeResult)

```

weightMoves

*Checks the compatibility of the data if the given poset is modified.***Description**

Checks the compatibility of the data if the given poset is modified. The resulting weight matrix is used to favour proposals with a higher compatibility with the data in the simulated annealing procedure.

Usage

```
weightMoves(X, P)
```

Arguments

X	The observed genotypes as a data frame or matrix.
P	The poset to be modified as a matrix.

Details

The allowed changes to the poset are additions and deletions. Direction inversion can occur as a sequence of deletion and addition. If an addition is not a cover relation, it is checked if there is only one intermediate state which is then removed, e.g. $P1 \rightarrow P2 \rightarrow P3$ will lead to $P1 \rightarrow P2$ and $P1 \rightarrow P3$. Changes resulting in a cyclic graph and connections on the diagonal give a weight of zero.

Value

Matrix with the compatibility weights at each position in the poset.

Note

Code adapted from the h-cbn program written in C by N. Beerenwinkel, M. Gerstung.

Author(s)

Patrick Knupfer

Examples

```
X <- testData[, -1]
P <- testPoset
weightMoves(X, P)
```

zdv

Zidovudine phenosense data.

Description

Consists of 7 resistance relevant mutations and the log10 fold-change against ZDV.

Usage

```
data(zdv)
```

Format

A data frame with 617 observations on the following 8 variables.

log10FC(ZDV) log10(FC) against ZDV.
41L Indicator for protease mutation 41L.
67N Indicator for protease mutation 67N.
69D Indicator for protease mutation 69D.
70R Indicator for protease mutation 70R.
210W Indicator for protease mutation 210W.
215Y Indicator for protease mutation 215Y.
219Q Indicator for protease mutation 219Q.

Examples

```
str(zdv)
```

Index

*Topic **datasets**

testData, [36](#)
testPoset, [37](#)
testResponsibilities, [37](#)
zdv, [41](#)

*Topic **package**

icbn-package, [1](#)

*Topic **utilities**

avgPheno, [2](#)
checkCompatibility, [3](#)
comparePosets, [3](#)
computeDistanceMatrix, [4](#)
constructPoset, [5](#)
convert2ConstraintMatrix, [6](#)
convert2EdgeList, [7](#)
convertRest2Poset, [7](#)
cvIcbn, [8](#)
doAnneal, [9](#)
doEm, [10](#)
estimateEpsilon, [11](#)
estimateMu, [12](#)
estimateSigma, [13](#)
estimateSS, [13](#)
estimateThetas, [14](#)
exitSet, [15](#)
fitCbn, [16](#)
generateData, [17](#)
generateMus, [18](#)
generateThetas, [19](#)
getGenotypeIdx, [20](#)
guessMus, [21](#)
guessParams, [22](#)
guessThetas, [23](#)
hammingDistance, [24](#)
hasse, [24](#)
hyperCube, [25](#)
hyperCubeNames, [26](#)
icbnNews, [26](#)
map2HyperCube, [27](#)
matrixPower, [27](#)
mixedRadixGeneration, [28](#)
orderIdeals, [29](#)
plotIsotonic, [30](#)

plotPoset, [31](#)
predictPheno, [32](#)
printParams, [33](#)
probCbn, [33](#)
probErr, [34](#)
probIso, [35](#)
transitiveClosure, [38](#)
tryEdge, [39](#)
weightMoves, [40](#)

avgPheno, [2](#)

checkCompatibility, [3](#)
comparePosets, [3](#)
computeDistanceMatrix, [4](#), [35](#)
constructPoset, [5](#)
convert2ConstraintMatrix, [6](#)
convert2EdgeList, [7](#)
convertRest2Poset, [7](#)
cvIcbn, [8](#)

doAnneal, [9](#)
doEm, [10](#)

estimateEpsilon, [11](#)
estimateMu, [12](#)
estimateSigma, [13](#)
estimateSS, [13](#)
estimateThetas, [14](#)
exitSet, [15](#)

fitCbn, [9](#), [16](#)

generateData, [17](#)
generateMus, [18](#)
generateThetas, [19](#)
getGenotypeIdx, [20](#)
guessMus, [21](#)
guessParams, [22](#)
guessThetas, [23](#)

hammingDistance, [24](#)
hasse, [24](#)
hyperCube, [25](#)
hyperCubeNames, [26](#)

`icbn` (*icbn-package*), 1
`icbn-package`, 1
`icbnNews`, 26

`map2HyperCube`, 27
`matrixPower`, 27
`mixedRadixGeneration`, 28

`orderIdeals`, 29

`plotIsotonic`, 30
`plotPoset`, 31
`predictPheno`, 16, 32
`printParams`, 33
`probCbn`, 33
`probErr`, 5, 34
`probIso`, 35

`testData`, 36
`testPoset`, 37
`testResponsibilities`, 37
`transitiveClosure`, 38
`tryEdge`, 39

`weightMoves`, 40

`zdv`, 41