

Package ‘RPMM’

November 16, 2009

Type Package

Title Recursively Partitioned Mixture Model

Version 1.05

Date 2009-11-16

Author E. Andres Houseman, Sc.D.

Maintainer E. Andres Houseman <E_Andres_Houseman@brown.edu>

Depends R (>= 2.3.1), cluster

Description Recursively Partitioned Mixture Model for Beta and Gaussian Mixtures. This is a model-based clustering algorithm that returns a hierarchy of classes, similar to hierarchical clustering, but also similar to finite mixture models.

License GPL (>= 2)

Repository CRAN

Date/Publication 2009-11-16 20:52:20

R topics documented:

betaEst	2
betaEstMultiple	3
betaObjf	4
blc	4
blcInitializeSplitDichotomizeUsingMean	5
blcInitializeSplitEigen	6
blcInitializeSplitFanny	6
blcInitializeSplitHClust	7
blcSplit	8
blcSplitCriterionBIC	9
blcSplitCriterionBICICL	10
blcSplitCriterionJustRecordEverything	11
blcSplitCriterionLevelWtdBIC	12

blcSplitCriterionLRT	13
blcSubTree	14
blcTree	14
blcTreeApply	17
blcTreeLeafClasses	18
blcTreeLeafMatrix	18
blcTreeOverallBIC	19
gaussEstMultiple	19
glc	20
glcInitializeSplitEigen	20
glcInitializeSplitFanny	21
glcInitializeSplitHClust	22
glcSplit	22
glcSplitCriterionBIC	23
glcSplitCriterionBICICL	24
glcSplitCriterionJustRecordEverything	25
glcSplitCriterionLevelWtdBIC	26
glcSplitCriterionLRT	27
glcSubTree	28
glcTree	28
glcTreeApply	31
glcTreeLeafClasses	32
glcTreeLeafMatrix	32
glcTreeOverallBIC	33
IlluminaMethylation	33
plot.blcTree	34
plot.glcTree	34
plotImage.blcTree	35
plotImage.glcTree	36
plotTree.blcTree	37
plotTree.glcTree	37
print.blcTree	38
print.glcTree	39

Index **40**

betaEst

Beta Distribution Maximum Likelihood Estimator

Description

Estimates a beta distribution via Maximum Likelihood

Usage

betaEst(y, w, weights)

Arguments

<code>y</code>	data vector
<code>w</code>	posterior weights
<code>weights</code>	case weights

Details

Typically not be called by user.

Value

(a,b) parameters

`betaEstMultiple` *Beta Maximum Likelihood on a Matrix*

Description

Maximum likelihood estimator for beta model on matrix of values (columns having different, independent beta distributions)

Usage

```
betaEstMultiple(Y, weights = NULL)
```

Arguments

<code>Y</code>	data matrix
<code>weights</code>	case weights

Value

A list of beta parameters and BIC

 betaObjf

Beta Maximum Likelihood Objective Function

Description

Objective function for fitting a beta model using maximum likelihood

Usage

```
betaObjf(logab, ydata, wdata, weights)
```

Arguments

logab	log(a,b) parameters
ydata	data vector
wdata	posterior weights
weights	case weights

Details

Typically not be called by user.

Value

negative log-likelihood

 blc

Beta Latent Class Model

Description

Fits a beta mixture model for any number of classes

Usage

```
blc(Y, w, maxiter = 25, tol = 1e-06, weights = NULL, verbose = TRUE)
```

Arguments

Y	Data matrix (n x j) on which to perform clustering
w	Initial weight matrix (n x k) representing classification
maxiter	Maximum number of EM iterations
tol	Convergence tolerance
weights	Case weights
verbose	Verbose output?

Details

Typically not be called by user.

Value

A list of parameters representing mixture model fit, including posterior weights and log-likelihood

`blcInitializeSplitDichotomizeUsingMean`
Initialize Gaussian Latent Class via Mean Dichotomization

Description

Creates a function for initializing latent class model by dichotomizing via mean over all responses

Usage

```
blcInitializeSplitDichotomizeUsingMean(threshold = 0.5, fuzz = 0.95)
```

Arguments

<code>threshold</code>	Mean threshold for determining class
<code>fuzz</code>	“fuzz” factor for producing imperfectly clustered subjects

Details

Creates a function $f(x)$ that will take a data matrix x and initialize a weight matrix for a two-class latent class model. Here, a simple threshold will be applied to the mean over all item responses. See [blcTree](#) for example of using “`blcInitializeSplit...`” to create starting values.

Value

A function $f(x)$ (see Details.)

See Also

[glcInitializeSplitFanny](#), [glcInitializeSplitHClust](#)

```
blcInitializeSplitEigen
```

Initialize Gaussian Latent Class via Eigendecomposition

Description

Creates a function for initializing latent class model based on Eigendecomposition

Usage

```
blcInitializeSplitEigen(eigendim = 1,
  assignmentf = function(s) (rank(s) - 0.5)/length(s))
```

Arguments

`eigendim` How many eigenvalues to use
`assignmentf` assignment function for transforming eigenvector to weight

Details

Creates a function $f(x)$ that will take a data matrix x and initialize a weight matrix for a two-class latent class model. Here, the initialized classes will be based on eigendecomposition of the variance of x . See [blcTree](#) for example of using “blcSplitCriterion...” to control split.

Value

A function $f(x)$ (see Details.)

See Also

[blcInitializeSplitDichotomizeUsingMean](#), [glcInitializeSplitFanny](#), [glcInitializeSplitHC](#)

```
blcInitializeSplitFanny
```

Initialize Beta Latent Class via Fanny

Description

Creates a function for initializing latent class model using the fanny algorithm

Usage

```
blcInitializeSplitFanny(nu = 2, nufac = 0.875, metric = "euclidean")
```

Arguments

<code>nu</code>	memb.exp parameter in fanny
<code>nufac</code>	Factor by which to multiply <code>nu</code> if an error occurs
<code>metric</code>	Metric to use for fanny

Details

Creates a function $f(x)$ that will take a data matrix x and initialize a weight matrix for a two-class latent class model. Here, the “fanny” algorithm will be used. See [blcTree](#) for example of using “blcSplitCriterion...” to control split.

Value

A function $f(x)$ (see Details.)

See Also

[blcInitializeSplitDichotomizeUsingMean](#), [blcInitializeSplitEigen](#), [blcInitializeSplitHC](#)

`blcInitializeSplitHClust`

Initialize Beta Latent Class via Hierarchical Clustering

Description

Creates a function for initializing latent class model using hierarchical clustering.

Usage

```
blcInitializeSplitHClust(metric = "manhattan", method = "ward")
```

Arguments

<code>metric</code>	Dissimilarity metric used for hierarchical clustering
<code>method</code>	Linkage method used for hierarchical clustering

Details

Creates a function $f(x)$ that will take a data matrix x and initialize a weight matrix for a two-class latent class model. Here, a two-branch split from hierarchical clustering will be used. See [blcTree](#) for example of using “blcSplitCriterion...” to control split.

Value

A function $f(x)$ (see Details.)

See Also

[blcInitializeSplitDichotomizeUsingMean](#), [blcInitializeSplitEigen](#), [blcInitializeSplitFa](#)

blcSplit

*Beta Latent Class Splitter***Description**

Splits a data set into two via a beta mixture model

Usage

```
blcSplit(x, initFunctions, weight = NULL, index = NULL, level = NULL,
        wthresh = 1e-09, verbose = TRUE, nthresh = 5,
        splitCriterion = NULL)
```

Arguments

x	Data matrix (n x j) on which to perform clustering
initFunctions	List of functions of type “blcInitialize...” for initializing latent class model. See blcInitializeFanny for an example of arguments and return values.
weight	Weight corresponding to the indices passed (see index). Defaults to 1 for all indices
index	Row indices of data matrix to include. Defaults to all (1 to n).
level	~~Describe level here~~
wthresh	Weight threshold for filtering data to children. Indices having weight less than this value will not be passed to children nodes.
verbose	Level of verbosity. Default=2 (too much). 0 for quiet.
nthresh	Total weight in node required for node to be a candidate for splitting. Nodes with weight less than this value will never split.
splitCriterion	Function of type “blcSplitCriterion...” for determining whether split should occur. See blcSplitCriterionBIC for an example of arguments and return values. Default behavior is blcSplitCriterionBIC (though the function is bypassed by internal calculations for some modest computational efficiency gains).

Details

Should not be called by user.

Value

A list of objects representing split.

 blcSplitCriterionBIC

Beta RPMM Split Criterion: Use BIC

Description

Split criterion function: compare BICs to determine split.

Usage

```
blcSplitCriterionBIC(llike1, llike2, weight, ww, J, level)
```

Arguments

llike1	one-class likelihood.
llike2	two-class likelihood.
weight	weights from RPMM node.
ww	“ww” from RPMM node.
J	Number of items.
level	Node level.

Details

This is a function of the form “glcSplitCriterion...”, which is required to return a list with at least a boolean value `split`, along with supporting information. See [blcTree](#) for example of using “blcSplitCriterion...” to control split.

Value

bic1	one-class (weighted) BIC
bic2	two-class (weighted) BIC
split	TRUE=split the node, FALSE=do not split the node.

See Also

[blcSplitCriterionBIC](#), [blcSplitCriterionJustRecordEverything](#), [blcSplitCriterionLevelWt](#),
[blcSplitCriterionLRT](#)

```
blcSplitCriterionBICICL
```

Beta RPMM Split Criterion: Use ICL-BIC

Description

Split criterion function: compare ICL-BICs to determine split (i.e. include entropy term in comparison).

Usage

```
blcSplitCriterionBICICL(llike1, llike2, weight, ww, J, level)
```

Arguments

<code>llike1</code>	one-class likelihood.
<code>llike2</code>	two-class likelihood.
<code>weight</code>	weights from RPMM node.
<code>ww</code>	“ww” from RPMM node.
<code>J</code>	Number of items.
<code>level</code>	Node level.

Details

This is a function of the form “`glcSplitCriterion...`”, which is required to return a list with at least a boolean value `split`, along with supporting information. See [blcTree](#) for example of using “`blcSplitCriterion...`” to control split.

Value

<code>bic1</code>	one-class (weighted) BIC
<code>bic2</code>	two-class (weighted) BIC
<code>entropy</code>	two-class entropy
<code>split</code>	TRUE=split the node, FALSE=do not split the node.

See Also

[blcSplitCriterionBICICL](#), [blcSplitCriterionJustRecordEverything](#), [blcSplitCriterionLevel](#), [blcSplitCriterionLRT](#)

```
blcSplitCriterionJustRecordEverything
```

Beta RPMM Split Criterion: Always Split and Record Everything

Description

Split criterion function: always split, but record everything as you go.

Usage

```
blcSplitCriterionJustRecordEverything(llike1, llike2, weight, ww, J, level)
```

Arguments

llike1	one-class likelihood.
llike2	two-class likelihood.
weight	weights from RPMM node.
ww	“ww” from RPMM node.
J	Number of items.
level	Node level.

Details

This is a function of the form “glcSplitCriterion...”, which is required to return a list with at least a boolean value `split`, along with supporting information. This function ALWAYS returns `split=TRUE`. Useful for gathering information. It is recommended that you set the `maxlev` argument in the main function to something less than infinity (say, 3 or 4). See [blcTree](#) for example of using “blcSplitCriterion...” to control split.

Value

llike1	Just returns llike1
llike2	Just returns llike2
J	Just returns J
weight	Just returns weight
ww	Just returns ww
degFreedom	Degrees-of-freedom for LRT
chiSquareStat	Chi-square statistic
split	TRUE=split the node, FALSE=do not split the node.

See Also

[blcSplitCriterionBIC](#), [blcSplitCriterionBICICL](#), [blcSplitCriterionLevelWtdBIC](#), [blcSplitCriterionLRT](#)

```
blcSplitCriterionLevelWtdBIC
```

Beta RPMM Split Criterion: Level-Weighted BIC

Description

Split criterion function: use a level-weighted version of BIC to determine split; there is an additional penalty incorporated for deep recursion.

Usage

```
blcSplitCriterionLevelWtdBIC(llike1, llike2, weight, ww, J, level)
```

Arguments

<code>llike1</code>	one-class likelihood.
<code>llike2</code>	two-class likelihood.
<code>weight</code>	weights from RPMM node.
<code>ww</code>	“ww” from RPMM node.
<code>J</code>	Number of items.
<code>level</code>	Node level.

Details

This is a function of the form “`glcSplitCriterion...`”, which is required to return a list with at least a boolean value `split`, along with supporting information. See [blcTree](#) for example of using “`blcSplitCriterion...`” to control split.

Value

<code>bic1</code>	One-class BIC, with additional penalty for deeper levels
<code>bic2</code>	Two-class BIC, with additional penalty for deeper levels
<code>split</code>	TRUE=split the node, FALSE=do not split the node.

See Also

[blcSplitCriterionBIC](#), [blcSplitCriterionBICICL](#), [blcSplitCriterionJustRecordEverything](#), [blcSplitCriterionLRT](#)

 blcSplitCriterionLRT

Beta RPMM Split Criterion: use likelihood ratio test p value

Description

Split criterion function: Use likelihood ratio test p value to determine split.

Usage

```
blcSplitCriterionLRT(llike1, llike2, weight, ww, J, level)
```

Arguments

llike1	one-class likelihood.
llike2	two-class likelihood.
weight	weights from RPMM node.
ww	“ww” from RPMM node.
J	Number of items.
level	Node level.

Details

This is a function of the form “blcSplitCriterion...”, which is required to return a list with at least a boolean value `split`, along with supporting information. See [blcTree](#) for example of using “blcSplitCriterion...” to control split.

Value

llike1	Just returns llike1
llike2	Just returns llike2
J	Just returns J
weight	Just returns weight
degFreedom	Degrees-of-freedom for LRT
chiSquareStat	Chi-square statistic
split	TRUE=split the node, FALSE=do not split the node.

See Also

[blcSplitCriterionBIC](#), [blcSplitCriterionBICICL](#), [blcSplitCriterionJustRecordEverything](#), [blcSplitCriterionLevelWtdBIC](#)

blcSubTree	<i>Beta Subtree</i>
------------	---------------------

Description

Subsets a “blcTree” object, i.e. considers the tree whose root is a given node.

Usage

```
blcSubTree(tr, node)
```

Arguments

tr	“blcTree” object to subset
node	Name of node to make root.

Details

Typically not be called by user.

Value

A “blcTree” object whose root is the given node of tr

blcTree	<i>Beta RPMM Tree</i>
---------	-----------------------

Description

Performs beta latent class modeling using recursively-partitioned mixture model

Usage

```
blcTree(x, initFunctions = list(blcInitializeSplitFanny()),
  weight = NULL, index = NULL, wthresh = 1e-08, nodename = "root",
  maxlevel = Inf, verbose = 2, nthresh = 5, level = 0, env = NULL,
  unsplit = NULL, splitCriterion = blcSplitCriterionBIC)
```

Arguments

<code>x</code>	Data matrix (n x j) on which to perform clustering. Missing values are supported. All values should lie strictly between 0 and 1.
<code>initFunctions</code>	List of functions of type “blcInitialize...” for initializing latent class model. See <code>blcInitializeFanny</code> for an example of arguments and return values.
<code>weight</code>	Weight corresponding to the indices passed (see <code>index</code>). Defaults to 1 for all indices
<code>index</code>	Row indices of data matrix to include. Defaults to all (1 to n).
<code>wthresh</code>	Weight threshold for filtering data to children. Indices having weight less than this value will not be passed to children nodes. Default=1E-8.
<code>nodename</code>	Name of object that will represent node in tree data object. Defaults to “root”. USER SHOULD NOT SET THIS.
<code>maxlevel</code>	Maximum depth to recurse. Default=Inf.
<code>verbose</code>	Level of verbosity. Default=2 (too much). 0 for quiet.
<code>nthresh</code>	Total weight in node required for node to be a candidate for splitting. Nodes with weight less than this value will never split. Defaults to 5.
<code>level</code>	Current level. Defaults to 0. USER SHUOLD NOT SET THIS.
<code>env</code>	Object of class “blcTree” to store tree data. Defaults to a new object. USER SHOULD NOT SET THIS.
<code>unsplit</code>	Latent class parameters from parent, to store in current node. Defaults to NULL for root. This is used in plotting functions. USER SHOULD NOT SET THIS.
<code>splitCriterion</code>	Function of type “blcSplitCriterion...” for determining whether a node should be split. See <code>blcSplitCriterionBIC</code> for an example of arguments and return values.

Details

This function is called recursively by itself. Upon each recursion, certain arguments (e.g. `nodename`) are reset. Do not attempt to set these arguments yourself.

Value

An object of class “blcTree”. This is an environment, each of whose component objects represents a node in the tree.

Note

The class “blcTree” is currently implemented as an environment object with nodes represented flatly, with name indicating position in hierarchy (e.g. “rLLR” = “right child of left child of left child of root”) This implementation is to make certain plotting and update functions simpler than would be required if the data were stored in a more natural “list of list” format.

The following error may appear during the course of the algorithm:

```
Error in optim(logab, betaObjf, ydata = y, wdata = w, weights = weights, :
  non-finite value supplied by optim
```

This is merely an indication that the node being split is too small, in which case the splitting will terminate at that node; in other words, it is nothing to worry about.

Author(s)

E. Andres Houseman

References

Houseman et al., Model-based clustering of DNA methylation array data: a recursive-partitioning algorithm for high-dimensional data arising as a mixture of beta distributions. *BMC Bioinformatics* 9:365, 2008.

See Also

[glcTree](#)

Examples

```
## Not run:
data(IlluminaMethylation)

heatmap(IllumBeta, scale="n",
        col=colorRampPalette(c("yellow", "black", "blue"), space="Lab")(128))

# Fit Gaussian RPMM
rpmm <- blcTree(IllumBeta, verbose=0)
rpmm

# Get weight matrix and show first few rows
rpmmWeightMatrix <- blcTreeLeafMatrix(rpmm)
rpmmWeightMatrix[1:3,]

# Get class assignments and compare with tissue
rpmmClass <- blcTreeLeafClasses(rpmm)
table(rpmmClass, tissue)

# Plot fit
par(mfrow=c(2,2))
plot(rpmm) ; title("Image of RPMM Profile")
plotTree.blcTree(rpmm) ; title("Dendrogram with Labels")
plotTree.blcTree(rpmm,
  labelFunction=function(u, digits) table(as.character(tissue[u$index])))
title("Dendrogram with Tissue Counts")

# Alternate initialization
rpmm2 <- blcTree(IllumBeta, verbose=0,
  initFunctions=list(blcInitializeSplitEigen(),
```

```

                                blcInitializeSplitFanny(nu=2.5))
rpmm2

# Alternate split criterion
rpmm3 <- blcTree(IllumBeta, verbose=0, maxlev=3,
  splitCriterion=blcSplitCriterionLevelWtdBIC)
rpmm3

rpmm4 <- blcTree(IllumBeta, verbose=0, maxlev=3,
  splitCriterion=blcSplitCriterionJustRecordEverything)
rpmm4$rLL$splitInfo$llike1
rpmm4$rLL$splitInfo$llike2

## End(Not run)

```

blcTreeApply

Recursive Apply Function for Beta RPMM Objects

Description

Recursively applies a function down the nodes of a Gaussian RPMM tree.

Usage

```
blcTreeApply(tr, f, start = "root", terminalOnly = FALSE, asObject = TRUE, ...)
```

Arguments

tr	Tree object to recurse
f	Function to apply to every node
start	Starting node. Default = "root".
terminalOnly	TRUE=only terminal nodes, FALSE=all nodes.
asObject	TRUE: f accepts node as object. FALSE: f accepts node by node name and object name, f(nn,tr). In the latter case, f should be defined as <code>f <- function(nn,tree){...}</code> .
...	Additional arguments to pass to f

Value

A list of results; names of elements are names of nodes.

`blcTreeLeafClasses` *Posterior Class Assignments for Beta RPMM*

Description

Gets a vector of posterior class membership assignments for terminal nodes.

Usage

```
blcTreeLeafClasses(tr)
```

Arguments

`tr` Tree from which to create assignments.

Details

See [blcTree](#) for example.

Value

Vector of class assignments

See Also

[blcTreeLeafMatrix](#)

`blcTreeLeafMatrix` *Posterior Weight Matrix for Beta RPMM*

Description

Gets a matrix of posterior class membership weights for terminal nodes.

Usage

```
blcTreeLeafMatrix(tr, rounding = 3)
```

Arguments

`tr` Tree from which to create matrix.
`rounding` Digits to round.

Details

See [blcTree](#) for example.

Value

N x K matrix of posterior weights

See Also

[blcTreeLeafClasses](#)

`blcTreeOverallBIC` *Overall BIC for Entire RPMM Tree (Beta version)*

Description

Computes the BIC for the latent class model represented by terminal nodes

Usage

```
blcTreeOverallBIC(tr, ICL = FALSE)
```

Arguments

<code>tr</code>	Tree object on which to compute BIC
<code>ICL</code>	Include ICL entropy term?

Value

BIC or BIC-ICL.

`gaussEstMultiple` *Gaussian Maximum Likelihood on a Matrix*

Description

Maximum likelihood estimator for Gaussian model on matrix of values (columns having different, independent Gaussian distributions)

Usage

```
gaussEstMultiple(Y, weights = NULL)
```

Arguments

<code>Y</code>	data matrix
<code>weights</code>	case weights

Value

A list of beta parameters and BIC

glc

Gaussian Finite Mixture Model

Description

Fits a Gaussian mixture model for any number of classes

Usage

```
glc(Y, w, maxiter = 100, tol = 1e-06, weights = NULL, verbose = TRUE)
```

Arguments

Y	Data matrix (n x j) on which to perform clustering
w	Initial weight matrix (n x k) representing classification
maxiter	Maximum number of EM iterations
tol	Convergence tolerance
weights	Case weights
verbose	Verbose output?

Details

Typically not be called by user.

Value

A list of parameters representing mixture model fit, including posterior weights and log-likelihood

glcInitializeSplitEigen

Initialize Gaussian Latent Class via Eigendecomposition

Description

Creates a function for initializing latent class model based on Eigendecomposition

Usage

```
glcInitializeSplitEigen(eigendim = 1,
  assignmentf = function(s) (rank(s) - 0.5)/length(s))
```

Arguments

eigendim	How many eigenvalues to use
assignmentf	assignment function for transforming eigenvector to weight

Details

Creates a function $f(x)$ that will take a data matrix x and initialize a weight matrix for a two-class latent class model. Here, the initialized classes will be based on eigendecomposition of the variance of x . See [glcTree](#) for example of using “glcInitializeSplit...” to create starting values.

Value

A function $f(x)$ (see Details.)

See Also

[glcInitializeSplitFanny](#), [glcInitializeSplitHClust](#)

glcInitializeSplitFanny

Initialize Gaussian Latent Class via Fanny

Description

Creates a function for initializing latent class model using the fanny algorithm

Usage

```
glcInitializeSplitFanny(nu = 2, nufac = 0.875, metric = "euclidean")
```

Arguments

<code>nu</code>	<code>memb.exp</code> parameter in fanny
<code>nufac</code>	Factor by which to multiply <code>nu</code> if an error occurs
<code>metric</code>	Metric to use for fanny

Details

Creates a function $f(x)$ that will take a data matrix x and initialize a weight matrix for a two-class latent class model. Here, the “fanny” algorithm will be used. See [glcTree](#) for example of using “glcInitializeSplit...” to create starting values.

Value

A function $f(x)$ (see Details.)

See Also

[glcInitializeSplitEigen](#), [glcInitializeSplitHClust](#)

```
glcInitializeSplitHClust
```

Initialize Gaussian Latent Class via Hierarchical Clustering

Description

Creates a function for initializing latent class model using hierarchical clustering.

Usage

```
glcInitializeSplitHClust(metric = "manhattan", method = "ward")
```

Arguments

<code>metric</code>	Dissimilarity metric used for hierarchical clustering
<code>method</code>	Linkage method used for hierarchical clustering

Details

Creates a function $f(x)$ that will take a data matrix x and initialize a weight matrix for a two-class latent class model. Here, a two-branch split from hierarchical clustering will be used. See [glcTree](#) for example of using “glcInitializeSplit...” to create starting values.

Value

A function $f(x)$ (see Details.)

See Also

[glcInitializeSplitEigen](#), [glcInitializeSplitFanny](#)

```
glcSplit
```

Gaussian Latent Class Splitter

Description

Splits a data set into two via a Gaussian mixture models

Usage

```
glcSplit(x, initFunctions, weight = NULL, index = NULL, level =
0, wthresh = 1e-09, verbose = TRUE, nthresh = 5,
splitCriterion = glcSplitCriterionBIC)
```

Arguments

x	Data matrix (n x j) on which to perform clustering
initFunctions	List of functions of type “glcInitialize...” for initializing latent class model. See glcInitializeFanny for an example of arguments and return values.
weight	Weight corresponding to the indices passed (see index). Defaults to 1 for all indices
index	Row indices of data matrix to include. Defaults to all (1 to n).
level	~~Describe level here~~
wthresh	Weight threshold for filtering data to children. Indices having weight less than this value will not be passed to children nodes.
verbose	Level of verbosity. Default=2 (too much). 0 for quiet.
nthresh	Total weight in node required for node to be a candidate for splitting. Nodes with weight less than this value will never split.
splitCriterion	Function of type “glcSplitCriterion...” for determining whether split should occur. See glcSplitCriterionBIC for an example of arguments and return values.

Details

Should not be called by user.

Value

A list of objects representing split.

glcSplitCriterionBIC
Gaussian RPMM Split Criterion: Use BIC

Description

Split criterion function: compare BICs to determine split.

Usage

glcSplitCriterionBIC(llikel, llike2, weight, ww, J, level)

Arguments

llikel	one-class likelihood.
llike2	two-class likelihood.
weight	weights from RPMM node.
ww	“ww” from RPMM node.
J	Number of items.
level	Node level.

Details

This is a function of the form “`glcSplitCriterion...`”, which is required to return a list with at least a boolean value `split`, along with supporting information. See [glcTree](#) for example of using “`glcSplitCriterion...`” to control split.

Value

<code>bic1</code>	one-class (weighted) BIC
<code>bic2</code>	two-class (weighted) BIC
<code>split</code>	TRUE=split the node, FALSE=do not split the node.

See Also

[glcSplitCriterionBIC](#), [glcSplitCriterionJustRecordEverything](#), [glcSplitCriterionLevelWt](#), [glcSplitCriterionLRT](#)

`glcSplitCriterionBICICL`

Gaussian RPMM Split Criterion: Use ICL-BIC

Description

Split criterion function: compare ICL-BICs to determine split (i.e. include entropy term in comparison).

Usage

```
glcSplitCriterionBICICL(llike1, llike2, weight, ww, J, level)
```

Arguments

<code>llike1</code>	one-class likelihood.
<code>llike2</code>	two-class likelihood.
<code>weight</code>	weights from RPMM node.
<code>ww</code>	“ww” from RPMM node.
<code>J</code>	Number of items.
<code>level</code>	Node level.

Details

This is a function of the form “`glcSplitCriterion...`”, which is required to return a list with at least a boolean value `split`, along with supporting information. See [glcTree](#) for example of using “`glcSplitCriterion...`” to control split.

Value

bic1	one-class (weighted) BIC
bic2	two-class (weighted) BIC
entropy	two-class entropy
split	TRUE=split the node, FALSE=do not split the node.

See Also

[glcSplitCriterionBICICL](#), [glcSplitCriterionJustRecordEverything](#), [glcSplitCriterionLevel](#), [glcSplitCriterionLRT](#)

glcSplitCriterionJustRecordEverything

Gaussian RPMM Split Criterion: Always Split and Record Everything

Description

Split criterion function: always split, but record everything as you go.

Usage

```
glcSplitCriterionJustRecordEverything(llike1, llike2, weight, ww, J, level)
```

Arguments

llike1	one-class likelihood.
llike2	two-class likelihood.
weight	weights from RPMM node.
ww	“ww” from RPMM node.
J	Number of items.
level	Node level.

Details

This is a function of the form “glcSplitCriterion...”, which is required to return a list with at least a boolean value `split`, along with supporting information. This function ALWAYS returns `split=TRUE`. Useful for gathering information. It is recommended that you set the `maxlev` argument in the main function to something less than infinity (say, 3 or 4). See [glcTree](#) for example of using “glcSplitCriterion...” to control split.

Value

<code>llike1</code>	Just returns <code>llike1</code>
<code>llike2</code>	Just returns <code>llike2</code>
<code>J</code>	Just returns <code>J</code>
<code>weight</code>	Just returns <code>weight</code>
<code>ww</code>	Just returns <code>ww</code>
<code>degFreedom</code>	Degrees-of-freedom for LRT
<code>chiSquareStat</code>	Chi-square statistic
<code>split</code>	TRUE=split the node, FALSE=do not split the node.

See Also

[glcSplitCriterionBIC](#), [glcSplitCriterionBICICL](#), [glcSplitCriterionLevelWtdBIC](#), [glcSplitCriterionLRT](#)

`glcSplitCriterionLevelWtdBIC`

Gaussian RPMM Split Criterion: Level-Weighted BIC

Description

Split criterion function: use a level-weighted version of BIC to determine split; there is an additional penalty incorporated for deep recursion.

Usage

```
glcSplitCriterionLevelWtdBIC(llike1, llike2, weight, ww, J, level)
```

Arguments

<code>llike1</code>	one-class likelihood.
<code>llike2</code>	two-class likelihood.
<code>weight</code>	weights from RPMM node.
<code>ww</code>	“ww” from RPMM node.
<code>J</code>	Number of items.
<code>level</code>	Node level.

Details

This is a function of the form “`glcSplitCriterion...`”, which is required to return a list with at least a boolean value `split`, along with supporting information. See [glcTree](#) for example of using “`glcSplitCriterion...`” to control split.

Value

bic1	One-class BIC, with additional penalty for deeper levels
bic2	Two-class BIC, with additional penalty for deeper levels
split	TRUE=split the node, FALSE=do not split the node.

See Also

[glcSplitCriterionBIC](#), [glcSplitCriterionBICICL](#), [glcSplitCriterionJustRecordEverything](#), [glcSplitCriterionLRT](#)

glcSplitCriterionLRT

Gaussian RPMM Split Criterion: Use likelihood ratio test p value

Description

Split criterion function: use likelihood ratio test p value to determine split.

Usage

```
glcSplitCriterionLRT(llike1, llike2, weight, ww, J, level)
```

Arguments

llike1	one-class likelihood.
llike2	two-class likelihood.
weight	weights from RPMM node.
ww	“ww” from RPMM node.
J	Number of items.
level	Node level.

Details

This is a function of the form “glcSplitCriterion...”, which is required to return a list with at least a boolean value `split`, along with supporting information. See [glcTree](#) for example of using “glcSplitCriterion...” to control split.

Value

llike1	Just returns llike1
llike2	Just returns llike2
J	Just returns J
weight	Just returns weight
degFreedom	Degrees-of-freedom for LRT
chiSquareStat	Chi-square statistic
split	TRUE=split the node, FALSE=do not split the node.

See Also

[glcSplitCriterionBIC](#), [glcSplitCriterionBICICL](#), [glcSplitCriterionJustRecordEverything](#),
[glcSplitCriterionLevelWtdBIC](#)

glcSubTree	<i>Gaussian Subtree</i>
------------	-------------------------

Description

Subsets a “glcTree” object, i.e. considers the tree whose root is a given node.

Usage

```
glcSubTree(tr, node)
```

Arguments

tr	“glcTree” object to subset
node	Name of node to make root.

Details

Typically not be called by user.

Value

A “glcTree” object whose root is the given node of tr

glcTree	<i>Gaussian RPMM Tree</i>
---------	---------------------------

Description

Performs Gaussian latent class modeling using recursively-partitioned mixture model

Usage

```
glcTree(x, initFunctions = list(glcInitializeSplitFanny(nu=1.5)), weight = NULL, in
```

Arguments

<code>x</code>	Data matrix (n x j) on which to perform clustering. Missing values are supported.
<code>initFunctions</code>	List of functions of type “glcInitialize...” for initializing latent class model. See <code>glcInitializeFanny</code> for an example of arguments and return values.
<code>weight</code>	Weight corresponding to the indices passed (see <code>index</code>). Defaults to 1 for all indices
<code>index</code>	Row indices of data matrix to include. Defaults to all (1 to n).
<code>wthresh</code>	Weight threshold for filtering data to children. Indices having weight less than this value will not be passed to children nodes. Default=1E-8.
<code>nodename</code>	Name of object that will represent node in tree data object. Defaults to “root”. USER SHOULD NOT SET THIS.
<code>maxlevel</code>	Maximum depth to recurse. Default=Inf.
<code>verbose</code>	Level of verbosity. Default=2 (too much). 0 for quiet.
<code>nthresh</code>	Total weight in node required for node to be a candidate for splitting. Nodes with weight less than this value will never split. Defaults to 5.
<code>level</code>	Current level. Defaults to 0. USER SHUOLD NOT SET THIS.
<code>env</code>	Object of class “glcTree” to store tree data. Defaults to a new object. USER SHOULD NOT SET THIS.
<code>unsplit</code>	Latent class parameters from parent, to store in current node. Defaults to NULL for root. This is used in plotting functions. USER SHOULD NOT SET THIS.
<code>splitCriterion</code>	Function of type “glcSplitCriterion...” for determining whether a node should be split. See <code>glcSplitCriterionBIC</code> for an example of arguments and return values.

Details

This function is called recursively by itself. Upon each recursion, certain arguments (e.g. `nodename`) are reset. Do not attempt to set these arguments yourself.

Value

An object of class “glcTree”. This is an environment, each of whose component objects represents a node in the tree.

Note

The class “glcTree” is currently implemented as an environment object with nodes represented flatly, with name indicating position in hierarchy (e.g. “rLLR” = “right child of left child of left child of root”) This implementation is to make certain plotting and update functions simpler than would be required if the data were stored in a more natural “list of list” format.

The following error may appear during the course of the algorithm:

```
Error in optim(logab, betaObjf, ydata = y, wdata = w, weights = weights, :
  non-finite value supplied by optim
```

This is merely an indication that the node being split is too small, in which case the splitting will terminate at that node; in other words, it is nothing to worry about.

Author(s)

E. Andres Houseman

References

Houseman et al., Model-based clustering of DNA methylation array data: a recursive-partitioning algorithm for high-dimensional data arising as a mixture of beta distributions. *BMC Bioinformatics* 9:365, 2008.

See Also

[blcTree](#)

Examples

```
data(IlluminaMethylation)

## Not run:
heatmap(IllumBeta, scale="n",
        col=colorRampPalette(c("yellow", "black", "blue"), space="Lab")(128))

## End(Not run)

# Fit Gaussian RPMM
rpmm <- glcTree(IllumBeta, verbose=0)
rpmm

# Get weight matrix and show first few rows
rpmmWeightMatrix <- glcTreeLeafMatrix(rpmm)
rpmmWeightMatrix[1:3,]

# Get class assignments and compare with tissue
rpmmClass <- glcTreeLeafClasses(rpmm)
table(rpmmClass, tissue)

## Not run:
# Plot fit
par(mfrow=c(2,2))
plot(rpmm) ; title("Image of RPMM Profile")
plotTree.glcTree(rpmm) ; title("Dendrogram with Labels")
plotTree.glcTree(rpmm,
  labelFunction=function(u,digits) table(as.character(tissue[u$index])))
title("Dendrogram with Tissue Counts")
```

```

# Alternate initialization
rpmm2 <- glcTree(IllumBeta, verbose=0,
  initFunctions=list(glcInitializeSplitEigen(),
                    glcInitializeSplitFanny(nu=2.5)))
rpmm2

# Alternate split criterion
rpmm3 <- glcTree(IllumBeta, verbose=0, maxlev=3,
  splitCriterion=glcSplitCriterionLevelWtdBIC)
rpmm3

rpmm4 <- glcTree(IllumBeta, verbose=0, maxlev=3,
  splitCriterion=glcSplitCriterionJustRecordEverything)
rpmm4$rLL$splitInfo$l1like1
rpmm4$rLL$splitInfo$l1like2

## End(Not run)

```

glcTreeApply

Recursive Apply Function for Gaussian RPMM Objects

Description

Recursively applies a function down the nodes of a Gaussian RPMM tree.

Usage

```
glcTreeApply(tr, f, start = "root", terminalOnly = FALSE,
  asObject = TRUE, ...)
```

Arguments

<code>tr</code>	Tree object to recurse
<code>f</code>	Function to apply to every node
<code>start</code>	Starting node. Default = "root".
<code>terminalOnly</code>	TRUE=only terminal nodes, FALSE=all nodes.
<code>asObject</code>	TRUE: <code>f</code> accepts node as object. FALSE: <code>f</code> accepts node by node name and object name, <code>f(nn,tr)</code> . In the latter case, <code>f</code> should be defined as <code>f <- function(nn,tree){...}</code> .
<code>...</code>	Additional arguments to pass to <code>f</code>

Value

A list of results; names of elements are names of nodes.

`glcTreeLeafClasses` *Posterior Class Assignments for Gaussian RPMM*

Description

Gets a vector of posterior class membership assignments for terminal nodes.

Usage

```
glcTreeLeafClasses(tr)
```

Arguments

`tr` Tree from which to create assignments.

Details

See [glcTree](#) for example.

Value

Vector of class assignments

See Also

[glcTreeLeafMatrix](#)

`glcTreeLeafMatrix` *Posterior Weight Matrix for Gaussian RPMM*

Description

Gets a matrix of posterior class membership weights for terminal nodes.

Usage

```
glcTreeLeafMatrix(tr, rounding = 3)
```

Arguments

`tr` Tree from which to create matrix.
`rounding` Digits to round.

Details

See [glcTree](#) for example.

Value

N x K matrix of posterior weights

See Also

[glcTreeLeafClasses](#)

glcTreeOverallBIC *Overall BIC for Entire RPMM Tree (Gaussian version)*

Description

Computes the BIC for the latent class model represented by terminal nodes

Usage

```
glcTreeOverallBIC(tr, ICL = FALSE)
```

Arguments

tr	Tree object on which to compute BIC
ICL	Include ICL entropy term?

Value

BIC or BIC-ICL.

illuminaMethylation
DNA Methylation Data for Normal Tissue Types

Description

Illumina GoldenGate DNA methylation data for 217 normal tissues. 100 most variable CpG sites.

Usage

```
illuminaMethylation
```

Format

a 217 x 100 matrix containing Illumina Avg Beta values (IllumBeta), and a corresponding factor vector of 217 tissue types (tissue).

References

Christensen BC, Houseman EA, et al. 2009 Aging and Environmental Exposures Alter Tissue-Specific DNA Methylation Dependent upon CpG Island Context. PLoS Genet 5(8): e1000602.

plot.blcTree *Plot a Beta RPMM Tree Profile*

Description

Plot method for objects of type “blcTree”. Plots profiles of terminal nodes in color. Method wrapper for `plotImage.blcTree`.

Usage

```
## S3 method for class 'blcTree':  
plot(x, ...)
```

Arguments

x RPMM object to plot.
... Additional arguments to pass to `plotImage.blcTree`.

Details

See [blcTree](#) for example.

plot.glcTree *Plot a Gaussian RPMM Tree Profile*

Description

Plot method for objects of type “glcTree”. Plots profiles of terminal nodes in color. Method wrapper for `plotImage.glcTree`.

Usage

```
## S3 method for class 'glcTree':  
plot(x, ...)
```

Arguments

x RPMM object to plot.
... Additional arguments to pass to `plotImage.glcTree`.

Details

See [glcTree](#) for example.

plotImage.blcTree *Plot a Beta RPMM Tree Profile*

Description

Plots profiles of terminal nodes in color.

Usage

```
plotImage.blcTree(env,  
  start = "r", method = "weight",  
  palette = colorRampPalette(c("yellow", "black", "blue"), space = "Lab")(128),  
  divcol = "red", xorder = NULL, dimensions = NULL, labelType = "LR")
```

Arguments

env	RPMM object to plot.
start	Node to plot (usually root)
method	Method to determine width of columns that represent classes: “weight” (subject weight in class) or dQuotebinary (depth in tree).
palette	Color palette to use for image plot.
divcol	Divider color
xorder	Order of variables. Can be useful for constant ordering across multiple plots.
dimensions	Subset of dimensions of source data to show. Defaults to all. Useful to show a subset of dimensions.
labelType	Label name type: “LR” or “01”.

Details

See [blcTree](#) for example.

Value

Returns a vector of indices similar to the `order` function, representing the orrdering of items used in the plot. This is useful for replicating the order in another plot, or for axis labeling.

plotImage.glcTree *Plot a Gaussian RPMM Tree Profile*

Description

Plots profiles of terminal nodes in color.

Usage

```
plotImage.glcTree(env,
  start = "r", method = "weight",
  palette = colorRampPalette(c("yellow", "black", "blue"), space = "Lab")(128),
  divcol = "red", xorder = NULL, dimensions = NULL, labelType = "LR", muColorEps =
```

Arguments

env	RPMM object to print.
start	Node to plot (usually root)
method	Method to determine width of columns that represent classes: “weight” (subject weight in class) or dQuotebinary (depth in tree).
palette	Color palette to use for image plot.
divcol	Divider color
xorder	Order of variables. Can be useful for constant ordering across multiple plots.
dimensions	Subset of dimensions of source data to show. Defaults to all. Useful to show a subset of dimensions.
labelType	Label name type: “LR” or “01”.
muColorEps	Small value to stabilize color generation.

Details

See [glcTree](#) for example.

Value

Returns a vector of indices similar to the `order` function, representing the orrdering of items used in the plot. This is useful for replicating the order in another plot, or for axis labeling.

plotTree.blcTree *Plot a Beta RPMM Tree Dendrogram*

Description

Alternate plot function for objects of type blcTree: plots a dendrogram

Usage

```
plotTree.blcTree(env, start = "r", labelFunction = NULL,
  buff = 4, cex = 0.9, square = TRUE, labelAllNodes = FALSE, labelDigits = 1, ...)
```

Arguments

env	Tree object to print
start	Note from which to start. Default="r" for "root".
labelFunction	Function for generating node labels. Useful for labeling each node with a value.
buff	Buffer for placing tree in plot window.
cex	Text size
square	Square dendrogram or "V" shaped
labelAllNodes	TRUE=All nodes will be labeled; FALSE=Terminal nodes only.
labelDigits	Digits to include in labels, if labelFunction returns numeric values.
...	Other parameters to be passed to labelFunction.

Details

This plots a dendrogram based on RPMM tree, with labels constructed from summaries of tree object. See [blcTree](#) for example.

plotTree.glcTree *Plot a Gaussian RPMM Tree Dendrogram*

Description

Alternate plot function for objects of type glcTree: plots a dendrogram

Usage

```
plotTree.glcTree(env, start = "r", labelFunction = NULL,
  buff = 4, cex = 0.9, square = TRUE, labelAllNodes = FALSE, labelDigits = 1, ...)
```

Arguments

<code>env</code>	Tree object to print
<code>start</code>	Note from which to start. Default="r" for "root".
<code>labelFunction</code>	Function for generating node labels. Useful for labeling each node with a value.
<code>buff</code>	Buffer for placing tree in plot window.
<code>cex</code>	Text size
<code>square</code>	Square dendrogram or "V" shaped
<code>labelAllNodes</code>	TRUE=All nodes will be labeled; FALSE=Terminal nodes only.
<code>labelDigits</code>	Digits to include in labels, if <code>labelFunction</code> returns numeric values.
<code>...</code>	Other parameters to be passed to <code>labelFunction</code> .

Details

This plots a dendrogram based on RPMM tree, with labels constructed from summaries of tree object. See [glcTree](#) for example.

```
print.blcTree      Print a Beta RPMM object
```

Description

Print method for objects of type `blcTree`

Usage

```
## S3 method for class 'blcTree':
print(x, ...)
```

Arguments

<code>x</code>	RPMM object to print
<code>...</code>	(Unused).

Details

See [blcTree](#) for example.

`print.glcTree` *Print a Gaussian RPMM object*

Description

Print method for objects of type `blcTree`

Usage

```
## S3 method for class 'glcTree':  
print(x, ...)
```

Arguments

<code>x</code>	RPMM object to print
<code>...</code>	(Unused).

Details

See [glcTree](#) for example.

Index

*Topic **cluster**

betaEst, 1
betaEstMultiple, 2
betaObjf, 3
blc, 3
blcInitializeSplitDichotomizeUsingMean, 4
blcInitializeSplitEigen, 5
blcInitializeSplitFanny, 5
blcInitializeSplitHClust, 6
blcSplit, 7
blcSplitCriterionBIC, 8
blcSplitCriterionBICICL, 9
blcSplitCriterionJustRecordEverything, 10
blcSplitCriterionLevelWtdBIC, 11
blcSplitCriterionLRT, 12
blcSubTree, 13
blcTree, 13
blcTreeApply, 16
blcTreeLeafClasses, 17
blcTreeLeafMatrix, 17
blcTreeOverallBIC, 18
gaussEstMultiple, 18
glc, 19
glcInitializeSplitEigen, 19
glcInitializeSplitFanny, 20
glcInitializeSplitHClust, 21
glcSplit, 21
glcSplitCriterionBIC, 22
glcSplitCriterionBICICL, 23
glcSplitCriterionJustRecordEverything, 24
glcSplitCriterionLevelWtdBIC, 25
glcSplitCriterionLRT, 26
glcSubTree, 27
glcTree, 27

glcTreeApply, 30
glcTreeLeafClasses, 31
glcTreeLeafMatrix, 31
glcTreeOverallBIC, 32
plot.blcTree, 33
plot.glcTree, 33
plotImage.blcTree, 34
plotImage.glcTree, 35
plotTree.blcTree, 36
plotTree.glcTree, 36
print.blcTree, 37
print.glcTree, 38

*Topic **datasets**

*Topic **tree**

blcInitializeSplitDichotomizeUsingMean, 4
blcInitializeSplitEigen, 5
blcInitializeSplitFanny, 5
blcInitializeSplitHClust, 6
blcSplit, 7
blcSplitCriterionBIC, 8
blcSplitCriterionBICICL, 9
blcSplitCriterionJustRecordEverything, 10
blcSplitCriterionLevelWtdBIC, 11
blcSplitCriterionLRT, 12
blcSubTree, 13
blcTree, 13
blcTreeApply, 16
blcTreeLeafClasses, 17
blcTreeLeafMatrix, 17
blcTreeOverallBIC, 18
glcInitializeSplitEigen, 19
glcInitializeSplitFanny, 20
glcInitializeSplitHClust, 21
glcSplit, 21
glcSplitCriterionBIC, 22

- glcSplitCriterionBICICL, 23
 - glcSplitCriterionJustRecordEverything, 20, 21
 - 24
 - glcSplitCriterionLevelWtdBIC, 25
 - glcSplitCriterionLRT, 26
 - glcSubTree, 27
 - glcTree, 27
 - glcTreeApply, 30
 - glcTreeLeafClasses, 31
 - glcTreeLeafMatrix, 31
 - glcTreeOverallBIC, 32
 - plot.blcTree, 33
 - plot.glcTree, 33
 - plotImage.blcTree, 34
 - plotImage.glcTree, 35
 - plotTree.blcTree, 36
 - plotTree.glcTree, 36
 - print.blcTree, 37
 - print.glcTree, 38
- betaEst, 1
- betaEstMultiple, 2
- betaObjf, 3
- blc, 3
- blcInitializeSplitDichotomizeUsingMean, 4, 5, 6
- blcInitializeSplitEigen, 5, 6
- blcInitializeSplitFanny, 5, 6
- blcInitializeSplitHClust, 6, 6
- blcSplit, 7
- blcSplitCriterionBIC, 8, 8, 10–12
- blcSplitCriterionBICICL, 9, 9–12
- blcSplitCriterionJustRecordEverything, 8, 9, 10, 11, 12
- blcSplitCriterionLevelWtdBIC, 8–10, 11, 12
- blcSplitCriterionLRT, 8–11, 12
- blcSubTree, 13
- blcTree, 4–6, 8–12, 13, 17, 29, 33, 34, 36, 37
- blcTreeApply, 16
- blcTreeLeafClasses, 17, 18
- blcTreeLeafMatrix, 17, 17
- blcTreeOverallBIC, 18
- gaussEstMultiple, 18
- glc, 19
- glcInitializeSplitEigen, 19, 20, 21
- glcInitializeSplitFanny, 4, 5, 20,
- 20, 21
- glcInitializeSplitHClust, 4, 5, 20,
- 21
- glcSplit, 21
- glcSplitCriterionBIC, 22, 23, 25–27
- glcSplitCriterionBICICL, 23, 24–27
- glcSplitCriterionJustRecordEverything, 23, 24, 24, 26, 27
- glcSplitCriterionLevelWtdBIC, 23,
- 24, 25, 25, 27
- glcSplitCriterionLRT, 23–25, 26, 26
- glcSubTree, 27
- glcTree, 15, 20, 21, 23–26, 27, 31, 33, 35,
- 37, 38
- glcTreeApply, 30
- glcTreeLeafClasses, 31, 32
- glcTreeLeafMatrix, 31, 31
- glcTreeOverallBIC, 32
- IllumBeta (*IlluminaMethylation*), 32
- IlluminaMethylation, 32
- plot.blcTree, 33
- plot.glcTree, 33
- plotImage.blcTree, 34
- plotImage.glcTree, 35
- plotTree.blcTree, 36
- plotTree.glcTree, 36
- print.blcTree, 37
- print.glcTree, 38
- tissue (*IlluminaMethylation*), 32