Package ‘RPMM’

February 19, 2015

Type Package
Title Recursively Partitioned Mixture Model
Version 1.20
Date 2014-09-13
Author E. Andres Houseman, Sc.D. and Devin C. Koestler, Ph.D.
Maintainer E. Andres Houseman <andres.houseman@oregonstate.edu>
Depends R (>= 2.3.12), 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)
NeedsCompilation no
Repository CRAN
Date/Publication 2014-09-14 07:22:44

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
betaEst

Beta Distribution Maximum Likelihood Estimator

Description

Estimates a beta distribution via Maximum Likelihood
**Usage**

\[ \text{betaEst}(y, w, \text{weights}) \]

**Arguments**

- \( y \)  
  data vector
- \( w \)  
  posterior weights
- \( \text{weights} \)  
  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**

\[ \text{betaEstMultiple}(Y, \text{weights} = \text{NULL}) \]

**Arguments**

- \( Y \)  
  data matrix
- \( \text{weights} \)  
  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**
```r
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**
```r
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

Usage

blcInitializeSplitDichotomizeUsingMean(threshold = 0.5, fuzz = 0.95)

Arguments

threshold Mean threshold for determining class
fuzz “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 glcTree 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**

```r
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`, `glcInitializeSplitHClust`

---

**blcInitializeSplitFanny**

*Initialize Beta Latent Class via Fanny*

**Description**

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

**Usage**

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

**Arguments**

- **nu**
  - memb.exp parameter in fanny
- **nufac**
  - Factor by which to multiply nu if an error occurs
- **metric**
  - 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`, `blcInitializeSplitHClust`

---

**blcInitializeSplitHClust**

*Initialize Beta Latent Class via Hierarchical Clustering*

**Description**

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

**Usage**

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

**Arguments**

- **metric**
  - Dissimilarity metric used for hierarchical clustering
- **method**
  - 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`, `blcInitializeSplitFanny`
**Description**

Splits a data set into two via a beta mixture model

**Usage**

```r
define(blcsplit)
  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` Current level.
- `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**

```r
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`, `blcSplitCriterionLevelWtdBIC`, `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

```r
blcSplitCriterionBICICL(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
- `entropy`: two-class entropy
- `split`: TRUE=split the node, FALSE=do not split the node.

#### See Also

- `blcSplitCriterionBICICL`
- `blcSplitCriterionJustRecordEverything`
- `blcSplitCriterionLevelWtdBIC`
- `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`
Beta RPM 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

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 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

blcSplitCriterionBIC, blcSplitCriterionBICICL, blcSplitCriterionJustRecordEverything, 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**

```r
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  Beta Subtree

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  Beta RPMM Tree

Description
Performs beta latent class modeling using recursively-partitioned mixture model.

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

x  Data matrix (n x j) on which to perform clustering. Missing values are supported. All values should lie strictly between 0 and 1.

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).

wthresh  Weight threshold for filtering data to children. Indices having weight less than this value will not be passed to children nodes. Default=1E-8.

nodename  Name of object that will represent node in tree data object. Defaults to “root”. USER SHOULD NOT SET THIS.

maxlevel  Maximum depth to recurse. Default=Inf.

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. Defaults to 5.

level  Current level. Defaults to 0. USER SHUOLD NOT SET THIS.

env  Object of class “blcTree” to store tree data. Defaults to a new object. USER SHOULD NOT SET THIS.

unsplit  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.

splitCriterion  Function of type “blcSplitCriterion...” for determining whether a node should be split. See blcSplitCriterionBIC 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


See Also

glcTree

Examples

```r
## 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)

# 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)))

# Alternate split criterion
```
**blctreeapply**  
Recursive Apply Function for Beta RPMM Objects

**Description**

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

**Usage**

```r
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 `f <- function(nn,tree){...}`.
- `...`  
  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
- tr: Tree object on which to compute BIC
- ICL: Include ICL entropy term?

Value
BIC or BIC-ICL.

---

ebayes Empirical Bayes predictions for a specific RPMM model

Description
Empirical Bayes predictions for a specific RPMM model

Usage
ebayes(rpmm, x, type, nodelist=NULL)

Arguments
- rpmm: RPMM object
- x: Data matrix
- type: RPMM type ("blc" or "glc")
- nodelist: RPMM subnode to use (default = root)
Details

Typically not be called by user.

Value

Matrix of empirical bayes predictions corresponding to \( x \).

---

**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**

- **Y**: data matrix
- **weights**: 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 (\( n \times j \)) on which to perform clustering
- **w**: Initial weight matrix (\( n \times k \)) 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

Description

Creates a function for initializing latent class model based on Eigendecomposition

Usage

\[
glcInitializeSplitEigen(eigendim = 1, 
  assignmentf = \text{function}(s) \ (\text{rank}(s) - 0.5)/\text{length}(s))
\]

Arguments

- \texttt{eigendim} : How many eigenvalues to use
- \texttt{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 \texttt{glcTree} for example of using “glcInitializeSplit...” to create starting values.

Value

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

See Also

\texttt{glcInitializeSplitFanny, glcInitializeSplitHClust}
### glcInitializeSplitFanny

**Initialize Gaussian Latent Class via Fanny**

**Description**

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

**Usage**

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

**Arguments**

- `nu` (memb.exp parameter in `fanny`)
- `nufac` (Factor by which to multiply `nu` if an error occurs)
- `metric` (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**

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

**Description**

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

**Arguments**

- **metric**
  Dissimilarity metric used for hierarchical clustering
- **method**
  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**

```r
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**
  Current level.
- **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(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

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

See Also

- glcSplitCriterionBIC, glcSplitCriterionJustRecordEverything, glcSplitCriterionLevelWtdBIC, glcSplitCriterionLRT
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
- 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
- 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, glcSplitCriterionLevelWtdBIC, glcSplitCriterionLRT
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
- `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=spli 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

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

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

\texttt{glcSplitCriterionLRT(llike1, llike2, weight, ww, J, level)}

Arguments

- \texttt{llike1}: one-class likelihood.
- \texttt{llike2}: two-class likelihood.
- \texttt{weight}: weights from RPMM node.
- \texttt{ww}: “ww” from RPMM node.
- \texttt{J}: Number of items.
- \texttt{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 \texttt{split}, along with supporting information. See \texttt{glcTree} for example of using “glcSplitCriterion...” to control split.

Value

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

See Also

\texttt{glcSplitCriterionBIC, glcSplitCriterionBICICL, glcSplitCriterionJustRecordEverything, glcSplitCriterionLevelWtdBIC}
glcSubTree

**Gaussian Subtree**

**Description**

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

**Usage**

```r
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

**Gaussian RPMM Tree**

**Description**

Performs Gaussian latent class modeling using recursively-partitioned mixture model

**Usage**

```r
glcTree(x, initFunctions = list(glcInitializeSplitFanny(nu=1.5)),
       weight = NULL, index = NULL, wthresh = 1e-08,
       nodename = "root", maxlevel = Inf, verbose = 2, nthresh = 5, level = 0,
       env = NULL, unsplit = NULL, splitCriterion = glcSplitCriterionBIC)
```
Arguments

x  Data matrix (n x j) on which to perform clustering. Missing values are supported.

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).

wthresh  Weight threshold for filtering data to children. Indices having weight less than this value will not be passed to children nodes. Default=1E-8.

nodeName  Name of object that will represent node in tree data object. Defaults to “root”. USER SHOULD NOT SET THIS.

maxlevel  Maximum depth to recurse. Default=Inf.

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. Defaults to 5.

level  Current level. Defaults to 0. USER SHOULD NOT SET THIS.

env  Object of class “glcTree” to store tree data. Defaults to a new object. USER SHOULD NOT SET THIS.

unsplit  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.

splitCriterion  Function of type “glcSplitCriterion...” for determining whether a node should be split. See glcSplitCriterionBIC 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**


**See Also**

blcTree

**Examples**

data(IlluminaMethylation)

```r
## 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[uIndex])))
title("Dendrogram with Tissue Counts")

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

Recursive Apply Function for Gaussian RPMM Objects

Description

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

Usage

```r
glcTreeApply(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 `f <- function(nn,tree){...}`.
- `...`: Additional arguments to pass to `f`

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

```r
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

```r
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.

glmLC Weighted GLM for latent class covariates

Description

Wrapper for glm function to incorporate weights corresponding to latent classes

Usage

`glmLC(y,W,family=quasibinomial(),eps=1e-8,Z=NULL)`

Arguments

- `y` outcome
- `W` weight matrix (rows=cases, # rows = length of y)
- `family` glm family (default = quasibinomial for logistic regression)
- `eps` threshold below which to delete pseudo-subject corresponding to a specific weight
- `Z` matrix of additional covariates
**Details**

This function is a wrapper for glm to incorporate weights corresponding to latent classes (e.g. from an RPMM prediction).

**Value**

a glm object

---

**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**


---

**llikerpmmobject**

*Data log-likelihood implied by a specific RPMM model*

**Description**

Data log-likelihood implied by a specific RPMM model

**Usage**

llikerpmmobject(o, x, type)

**Arguments**

- o: RPMM object
- x: Data matrix
- type: RPMM type ("bcl" or "glc")
plot.glcTree

Details
Typically not be called by user.

Value
Vector of loglikelihoods corresponding to rows of x.

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 \texttt{glcTree} for example.

\begin{verbatim}
plotImage.blcTree

Plot a Beta RPMM Tree Profile

Description

Plots profiles of terminal nodes in color.

Usage

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

Arguments

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

Details

See \texttt{blcTree} for example.

Value

Returns a vector of indices similar to the \texttt{order} function, representing the ordering of items used in the plot. This is useful for replicating the order in another plot, or for axis labeling.
Description

Plots profiles of terminal nodes in color.

Usage

```r
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 = 1e-08)
```

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 ordering 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,
                 buffer = 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.
buffer 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,
                 buffer = 4, cex = 0.9, square = TRUE, labelAllNodes = FALSE, labelDigits = 1, ...)
predict.blcTree

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 glcTree for example.

### predict.blcTree

**Predict using a Beta RPMM object**

Description

Prediction method for objects of type blcTree

Usage

```r
## S3 method for class 'blcTree'
predict(object, newdata=NULL, nodelist=NULL, type="weight",...)
```

Arguments

- **object**: RPMM object to print
- **newdata**: external data matrix from which to apply predictions
- **nodelist**: RPMM subnode to use (default = root)
- **type**: output type: "weight" produces output similar to blcTreeLeafMatrix, "class" produces output similar to blcTreeLeafClasses.
- **...**: (Unused).

Details

This function is similar to blcTreeLeafMatrix and blcTreeLeafClasses, except that it supports prediction on an external data set via the argument newdata.

See Also

- blcTreeLeafMatrix
**predict.glcTree**

`predict.glcTree` *Predict using a Gaussian RPMM object*

**Description**

Prediction method for objects of type glcTree

**Usage**

```
# S3 method for class 'glcTree'
predict(object, newdata=NULL, nodelist=NULL, type="weight", ...)
```

**Arguments**

- `object` RPMM object to print
- `newdata` external data matrix from which to apply predictions
- `nodelist` RPMM subnode to use (default = root)
- `type` output type: "weight" produces output similar to `glcTreeLeafMatrix`, "class" produces output similar to `glcTreeLeafClasses`.
  - `...` (Unused).

**Details**

This function is similar to `glcTreeLeafMatrix` and `glcTreeLeafClasses`, except that it supports prediction on an external data set via the argument `newdata`.

**See Also**

- `glcTreeLeafMatrix`

**print.blcTree**

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

**Description**

Print method for objects of type blcTree

**Usage**

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

x  RPMM object to print
...
(Unused).

Details

See blcTree for example.

---

print.glcTree  
*Print a Gaussian RPMM object*

---

Description

Print method for objects of type blcTree

Usage

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

Arguments

x  RPMM object to print
...
(Unused).

Details

See glcTree for example.
Index

*Topic cluster
  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, 20
  glc, 20
  glcInitializeSplitEigen, 21
  glcInitializeSplitFanny, 22
  glcInitializeSplitHClust, 22
  glcSplit, 23
  glcSplitCriterionBIC, 24
  glcSplitCriterionBICICL, 25
  glcSplitCriterionJustRecordEverything, 26
  glcSplitCriterionLevelWtdBIC, 27
  glcSplitCriterionLRT, 28
  glcSubTree, 29
  glcTree, 29
  glcTreeApply, 32
  glcTreeLeafClasses, 33
  glcTreeLeafMatrix, 33
  glcTreeOverallBIC, 34
  plot.blcTree, 36
  plot.glcTree, 36
  plotImage.blcTree, 37
  plotImage.glcTree, 38
  plotTree.blcTree, 39
  plotTree.glcTree, 39
  predict.blcTree, 40
  predict.glcTree, 41
  print.blcTree, 41
  print.glcTree, 42

*Topic datasets
  IlluminaMethylation, 35

*Topic tree
  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, 20
  glc, 20
  glcInitializeSplitEigen, 21
  glcInitializeSplitFanny, 22
  glcInitializeSplitHClust, 22
  glcSplit, 23
  glcSplitCriterionBIC, 24
  glcSplitCriterionBICICL, 25
  glcSplitCriterionJustRecordEverything, 26
  glcSplitCriterionLevelWtdBIC, 27
  glcSplitCriterionLRT, 28
  glcSubTree, 29
  glcTree, 29
  glcTreeApply, 32
  glcTreeLeafClasses, 33
  glcTreeLeafMatrix, 33
  glcTreeOverallBIC, 34
  plot.blcTree, 36
  plot.glcTree, 36
  plotImage.blcTree, 37
  plotImage.glcTree, 38
  plotTree.blcTree, 39
  plotTree.glcTree, 39
  predict.blcTree, 40
  predict.glcTree, 41
  print.blcTree, 41
  print.glcTree, 42
glcSplitCriterionJustRecordEverything, 26
glcSplitCriterionLevelWtdBIC, 27
glcSplitCriterionLRT, 28
glcSubTree, 29
glcTree, 29
glcTreeApply, 32
glcTreeLeafClasses, 33
glcTreeLeafMatrix, 33
glcTreeOverallBIC, 34
plot.blcTree, 36
plot.glcTree, 36
plotImage.blcTree, 37
plotImage.glcTree, 38
plotTree.blcTree, 39
plotTree.glcTree, 39
predict.blcTree, 40
predict.glcTree, 41
print.blcTree, 41
print.glcTree, 42

betaEst, 2
betaEstMultiple, 3
betaObjf, 4
blc, 4

blcInitializeSplitDichotomizeUsingMean, 5, 6, 7
blcInitializeSplitEigen, 6, 7
blcInitializeSplitFanny, 6, 7
blcInitializeSplitHClust, 7, 7
blcSplit, 8
blcSplitCriterionBIC, 9, 9, 11–13
blcSplitCriterionBICICL, 10, 10, 11–13
blcSplitCriterionJustRecordEverything, 9, 10, 11, 12, 13
blcSplitCriterionLevelWtdBIC, 9–11, 12, 13
blcSplitCriterionLRT, 9–12, 13
blcSubTree, 14
blcTree, 5–7, 9–13, 14, 18, 31, 36, 37, 39, 42
blcTreeApply, 17
blcTreeLeafClasses, 18, 19, 40
blcTreeLeafMatrix, 18, 18, 40
blcTreeOverallBIC, 19

ebayes, 19

gaussEstMultiple, 20
glc, 20

glcInitializeSplitEigen, 21, 22, 23
glcInitializeSplitFanny, 5, 6, 21, 22, 23
glcInitializeSplitHClust, 5, 6, 21, 22, 22
glcSplit, 23
glcSplitCriterionBIC, 24, 24, 26–28
glcSplitCriterionBICICL, 25, 25, 26–28
glcSplitCriterionJustRecordEverything, 24, 25, 26, 27, 28
glcSplitCriterionLevelWtdBIC, 24–26, 27, 28
glcSplitCriterionLRT, 24–27, 28
glcSubTree, 29
glcTree, 16, 21–28, 29, 33, 37, 38, 40, 42
glcTreeApply, 32
glcTreeLeafClasses, 33, 34, 41
glcTreeLeafMatrix, 33, 33, 41
glcTreeOverallBIC, 34
glmLC, 34

IllumBeta(IlluminaMethylation), 35
IlluminaMethylation, 35

llikeRPMMOObject, 35

plot.blcTree, 36
plot.glcTree, 36
plotImage.blcTree, 37
plotImage.glcTree, 38
plotTree.blcTree, 39
plotTree.glcTree, 39
predict.blcTree, 40
predict.glcTree, 41
print.blcTree, 41
print.glcTree, 42

tissue(IlluminaMethylation), 35

illuminamethylation, 35