Package 'dynamicTreeCut'

June 13, 2014

Version 1.62

Date 2014-05-07

Title Methods for detection of clusters in hierarchical clustering dendrograms.

Author
Peter Langfelder <Peter.Langfelder@gmail.com> and Bin Zhang <binzhang.ucla@gmail.com>,with c
tributions from Steve Horvath <SHorvath@mednet.ucla.edu>

Maintainer Peter Langfelder <Peter.Langfelder@gmail.com>

Depends R (>= 2.3.0), stats

ZipData no

License GPL (>= 2)

Description Contains methods for detection of clusters in hierarchical clustering dendrograms.

URL http://www.genetics.ucla.edu/labs/horvath/CoexpressionNetwork/BranchCutting/

R topics documented:

`dynamicTreeCut-package` *Methods for detection of clusters in hierarchical clustering dendrograms.*

Description

Contains methods for detection of clusters in hierarchical clustering dendrograms.

Details

Package:	dynamicTreeCut
Version:	1.62
Date:	2014-05-07
Depends:	R, stats
ZipData:	no
License:	GPL version 2 or newer
URL:	http://www.genetics.ucla.edu/labs/horvath/CoexpressionNetwork/BranchCutting/

Index:

Author(s)

Peter Langfelder <Peter.Langfelder@gmail.com> and Bin Zhang <binzhang.ucla@gmail.com>, with contributions from Steve Horvath <SHorvath@mednet.ucla.edu>

Maintainer: Peter Langfelder <Peter.Langfelder@gmail.com>

`cutreeDynamic` *Adaptive branch pruning of hierarchical clustering dendrograms.*

Description

This wrapper provides a common access point for two methods of adaptive branch pruning of hierarchical clustering dendrograms.

Usage

```
cutreeDynamic(
      dendro, cutHeight = NULL, minClusterSize = 20,

      # Basic tree cut options
      method = "hybrid", distM = NULL,
      deepSplit = (ifelse(method=="hybrid", 1, FALSE)),

      # Advanced options
      maxCoreScatter = NULL, minGap = NULL,
      maxAbsCoreScatter = NULL, minAbsGap = NULL,

      minSplitHeight = NULL, minAbsSplitHeight = NULL,

      # External (user-supplied) measure of branch split
      externalBranchSplitFnc = NULL, minExternalSplit = NULL,
      externalSplitOptions = list(),
      externalSplitFncNeedsDistance = NULL,
      assumeSimpleExternalSpecification = TRUE,

      # PAM stage options
      pamStage = TRUE, pamRespectsDendro = TRUE,
      useMedoids = FALSE, maxDistToLabel = NULL,
      maxPamDist = cutHeight,
      respectSmallClusters = TRUE,

      # Various options
      verbose = 2, indent = 0)
```

Arguments

`dendro`	A hierarchical clustering dendorgram such as one returned by `hclust`.
`cutHeight`	Maximum joining heights that will be considered. For `method=="tree"` it defaults to 0.99. For `method=="hybrid"` it defaults to 99% of the range between the 5th percentile and the maximum of the joining heights on the dendrogram.
`minClusterSize`	Minimum cluster size.
`method`	Chooses the method to use. Recognized values are "hybrid" and "tree".
`distM`	Only used for method "hybrid". The distance matrix used as input to `hclust`. If not given and `method == "hybrid"`, the function will issue a warning and default to `method = "tree"`.
`deepSplit`	For method "hybrid", can be either logical or integer in the range 0 to 4. For method "tree", must be logical. In both cases, provides a rough control over sensitivity to cluster splitting. The higher the value (or if `TRUE`), the more and smaller clusters will be produced. For the "hybrid" method, a finer control can be achieved via `maxCoreScatter` and `minGap` below.
`maxCoreScatter`	Only used for method "hybrid". Maximum scatter of the core for a branch to be a cluster, given as the fraction of `cutHeight` relative to the 5th percentile of

joining heights. See Details.

`minGap` Only used for method "hybrid". Minimum cluster gap given as the fraction of the difference between `cutHeight` and the 5th percentile of joining heights.

`maxAbsCoreScatter`
Only used for method "hybrid". Maximum scatter of the core for a branch to be a cluster given as absolute heights. If given, overrides `maxCoreScatter`.

`minAbsGap` Only used for method "hybrid". Minimum cluster gap given as absolute height difference. If given, overrides `minGap`.

`minSplitHeight`
Minimum split height given as the fraction of the difference between `cutHeight` and the 5th percentile of joining heights. Branches merging below this height will automatically be merged. Defaults to zero but is used only if `minAbsSplitHeight` below is `NULL`.

`minAbsSplitHeight`
Minimum split height given as an absolute height. Branches merging below this height will automatically be merged. If not given (default), will be determined from `minSplitHeight` above.

`externalBranchSplitFnc`
Optional function to evaluate split (dissimilarity) between two branches. Either a single function or a list in which each component is a function (see `assumeSimpleExternalSpecification` below for how to specify a single function). Each function can be specified by name (a character string) or the actual function object. Each given function must take arguments `branch1` and `branch2` that specify the indices of objects in the two branches whose dissimilarity is to be evaluated, and possibly other arguments. It must return a number that quantifies the dissimilarity of the two branches. The returned value will be compared to `minExternalSplit` (see below). This argument is only used for method "hybrid".

`minExternalSplit`
Thresholds to decide whether two branches should be merged. It should be a numeric vector of the same length as the number of functions in `externalBranchSplitFnc` above. Only used for method "hybrid".

`externalSplitOptions`
Further arguments to function `externalBranchSplitFnc`. If only one external function is specified in `externalBranchSplitFnc` above, `externalSplitOptions` can be a named list of arguments or a list with one component that is in turn the named list of further arguments for `externalBranchSplitFnc[[1]]`. The argument `assumeSimpleExternalSpecification` controls which of the two possibilities should be assumed. If multiple functions are specified in `externalBranchSplitFnc`, `externalSplitOptions` must be a list in which each component is a named list giving the further arguments for the corresponding function in `externalBranchSplitFnc`. Only used for method "hybrid".

`externalSplitFncNeedsDistance`
Optional specification of whether the external branch split functions need the distance matrix as one of their arguments. Either `NULL` or a logical vector with one element per branch split function that specifies whether the corresponding branch split function expects the distance matrix as one of its arguments. The default `NULL` is interpreted as a vector of `TRUE`. When dealing with a large number of objects, setting this argument to `FALSE` whenever possible can prevent unnecessary memory utilization.

`assumeSimpleExternalSpecification`
: Logical: when `minExternalSplit` above is a scalar (has length 1), should the function assume a simple specification of `externalBranchSplitFnc` and `externalSplitOptions`? If `TRUE`, `externalBranchSplitFnc` is taken as the function specification and `externalSplitOptions` the named list of options. This is suitable for simple direct calls of this function. If `FALSE`, `externalBranchSplitFnc` is assumed to be a list with a single component which specifies the function, and `externalSplitOptions` is a list with one component that is in turn the named list of further arguments for `externalBranchSplitFnc[[1]]`.

`pamStage`
: Only used for method "hybrid". If TRUE, the second (PAM-like) stage will be performed.

`pamRespectsDendro`
: Logical, only used for method "hybrid". If `TRUE`, the PAM stage will respect the dendrogram in the sense that objects and small clusters will only be assigned to clusters that belong to the same branch that the objects or small clusters being assigned belong to.

`useMedoids`
: Only used for method "hybrid" and only if `labelUnlabeled==TRUE`. If TRUE, the second stage will be use object to medoid distance; if FALSE, it will use average object to cluster distance. The default (FALSE) is recommended.

`maxDistToLabel`
: Deprecated, use `maxPamDist` instead. Only used for method "hybrid" and only if `labelUnlabeled==TRUE`. Maximum object distance to closest cluster that will result in the object assigned to that cluster.

`maxPamDist`
: Only used for method "hybrid" and only if `labelUnlabeled==TRUE`. Maximum object distance to closest cluster that will result in the object assigned to that cluster. Defaults to `cutHeight`.

`respectSmallClusters`
: Only used for method "hybrid" and only if `labelUnlabeled==TRUE`. If TRUE, branches that failed to be clusters in stage 1 only because of insufficient size will be assigned together in stage 2. If FALSE, all objects will be assigned individually.

`verbose`
: Controls the verbosity of the output. 0 will make the function completely quiet, values up to 4 gradually increase verbosity.

`indent`
: Controls indentation of printed messages (see `verbose` above). Each unit adds two spaces before printed messages; useful when several functions' output is to be nested.

Details

This is a wrapper for two related but different methods for cluster detection in hierarchical clustering dendrograms.

In order to make the shape parameters `maxCoreScatter` and `minGap` more universal, their values are interpreted relative to `cutHeight` and the 5th percetile of the merging heights (we arbitrarily chose the 5th percetile rather than the minimum for reasons of stability). Thus, the absolute maximum allowable core scatter is calculated as `maxCoreScatter * (cutHeight - refHeight) + refHeight` and the absolute minimum allowable gap as `minGap * (cutHeight - refHeight)`, where `refHeight` is the 5th percentile of the merging heights.

Value

A vector of numerical labels giving assignment of objects to modules. Unassigned objects are labeled 0, the largest module has label 1, next largest 2 etc.

Author(s)

Peter Langfelder, <Peter.Langfelder@gmail.com>

References

Langfelder P, Zhang B, Horvath S, 2007. http://www.genetics.ucla.edu/labs/horvath/CoexpressionNetwork/BranchCutting

See Also

hclust, cutreeHybrid, cutreeDynamicTree.

---

cutreeDynamicTree *Dynamic dendrogram pruning based on dendrogram only*

Description

Detect clusters in a hierarchical dendrogram using a variable cut height approach. Uses only the information in the dendrogram itself is used (which may give incorrect assignment for outlying objects).

Usage

```
cutreeDynamicTree(dendro, maxTreeHeight = 1, deepSplit = TRUE, minModuleSize = 5
```

Arguments

dendro	Hierarchical clustering dendrogram such produced by hclust.
maxTreeHeight	Maximum joining height of objects to be considered part of clusters.
deepSplit	If TRUE, method will favor sensitivity and produce more smaller clusters. When FALSE, there will be fewer bigger clusters.
minModuleSize	Minimum module size. Branches containing fewer than minModuleSize objects will be left unlabeled.

Details

A variable height branch pruning technique for dendrograms produced by hierarchical clustering. Initially, branches are cut off at the height maxTreeHeight; the resulting clusters are then examined for substructure and if subclusters are detected, they are assigned separate labels. Subclusters are detected by structure and are required to have a minimum of minModuleSize objects on them to be assigned a separate label. A rough degree of control over what it means to be a subcluster is implemented by the parameter deepSplit.

Value

A vector of numerical labels giving assignment of objects to modules. Unassigned objects are labeled 0, the largest module has label 1, next largest 2 etc.

Author(s)

Bin Zhang, <binzhang.ucla@gmail.com>, with contributions by Peter Langfelder, <Peter.Langfelder@gma

References

http://www.genetics.ucla.edu/labs/horvath/CoexpressionNetwork/BranchCutting

See Also

hclust, cutreeHybrid

---

cutreeHybrid *Hybrid adaptive tree cut for hierarchical clustering dendrograms.*

Description

Detect clusters in a dendorgram produced by the function hclust.

Usage

```
cutreeHybrid(
      # Input data: basic tree cutiing
      dendro, distM,

      # Branch cut criteria and options
      cutHeight = NULL, minClusterSize = 20, deepSplit = 1,

      # Advanced options
      maxCoreScatter = NULL, minGap = NULL,
      maxAbsCoreScatter = NULL, minAbsGap = NULL,

      minSplitHeight = NULL, minAbsSplitHeight = NULL,

      # External (user-supplied) measure of branch split
      externalBranchSplitFnc = NULL, minExternalSplit = NULL,
      externalSplitOptions = list(),
      externalSplitFncNeedsDistance = NULL,
      assumeSimpleExternalSpecification = TRUE,


      # PAM stage options
      pamStage = TRUE, pamRespectsDendro = TRUE,
      useMedoids = FALSE,
      maxPamDist = cutHeight,
      respectSmallClusters = TRUE,
```

```
        # Various options
        verbose = 2, indent = 0)
```

Arguments

dendro	a hierarchical clustering dendorgram such as one returned by hclust.
distM	Distance matrix that was used as input to hclust.
cutHeight	Maximum joining heights that will be considered. It defaults to 99 of the range between the 5th percentile and the maximum of the joining heights on the dendrogram.
minClusterSize	Minimum cluster size.
deepSplit	Either logical or integer in the range 0 to 4. Provides a rough control over sensitivity to cluster splitting. The higher the value, the more and smaller clusters will be produced. A finer control can be achieved via maxBranchCor, minBranchSplit, maxCoreScatter and minGap below.
maxCoreScatter	Maximum scatter of the core for a branch to be a cluster, given as the fraction of cutHeight relative to the 5th percentile of joining heights. See Details.
minGap	Minimum cluster gap given as the fraction of the difference between cutHeight and the 5th percentile of joining heights.
maxAbsCoreScatter	Maximum scatter of the core for a branch to be a cluster given as absolute heights. If given, overrides maxCoreScatter.
minAbsGap	Minimum cluster gap given as absolute height difference. If given, overrides minGap.
minSplitHeight	Minimum split height given as the fraction of the difference between cutHeight and the 5th percentile of joining heights. Branches merging below this height will automatically be merged. Defaults to zero but is used only if minAbsSplitHeight below is NULL.
minAbsSplitHeight	Minimum split height given as an absolute height. Branches merging below this height will automatically be merged. If not given (default), will be determined from minSplitHeight above.
externalBranchSplitFnc	Optional function to evaluate split (dissimilarity) between two branches. Either a single function or a list in which each component is a function (see assumeSimpleExternalSpecification below for how to specify a single function). Each function can be specified by name (a character string) or the actual function object. Each given function must take arguments branch1 and branch2 that specify the indices of objects in the two branches whose dissimilarity is to be evaluated, and possibly other arguments. It must return a number that quantifies the dissimilarity of the two branches. The returned value will be compared to minExternalSplit (see below). This argument is only used for method "hybrid".
minExternalSplit	Thresholds to decide whether two branches should be merged. It should be a numeric vector of the same length as the number of functions in externalBranchSplitFnc above. Only used for method "hybrid".

externalSplitOptions
: Further arguments to function externalBranchSplitFnc. If only one external function is specified in externalBranchSplitFnc above, externalSplitOptions can be a named list of arguments or a list with one component that is in turn the named list of further arguments for externalBranchSplitFnc[[1]]. The argument assumeSimpleExternalSpecification controls which of the two possibilities should be assumed. If multiple functions are specified in externalBranchSplitFnc, externalSplitOptions must be a list in which each component is a named list giving the further arguments for the corresponding function in externalBranchSplitFnc. Only used for method "hybrid".

externalSplitFncNeedsDistance
: Optional specification of whether the external branch split functions need the distance matrix as one of their arguments. Either NULL or a logical vector with one element per branch split function that specifies whether the corresponding branch split function expects the distance matrix as one of its arguments. The default NULL is interpreted as a vector of TRUE. When dealing with a large number of objects, setting this argument to FALSE whenever possible can prevent unnecessary memory utilization.

assumeSimpleExternalSpecification
: Logical: when minExternalSplit above is a scalar (has length 1), should the function assume a simple specification of externalBranchSplitFnc and externalSplitOptions? If TRUE, externalBranchSplitFnc is taken as the function specification and externalSplitOptions the named list of options. This is suitable for simple direct calls of this function. If FALSE, externalBranchSplitFnc is assumed to be a list with a single component which specifies the function, and externalSplitOptions is a list with one component that is in turn the named list of further arguments for externalBranchSplitFnc[[1]].

pamStage
: Logical, only used for method "hybrid". If TRUE, the second (PAM-like) stage will be performed.

pamRespectsDendro
: Logical, only used for method "hybrid". If TRUE, the PAM stage will respect the dendrogram in the sense an object can be PAM-assigned only to clusters that lie below it on the branch that the object is merged into. See cutreeDynamic for more details.

useMedoids
: if TRUE, the second stage will be use object to medoid distance; if FALSE, it will use average object to cluster distance. The default (FALSE) is recommended.

maxPamDist
: Maximum object distance to closest cluster that will result in the object assigned to that cluster. Defaults to cutHeight.

respectSmallClusters
: If TRUE, branches that failed to be clusters in stage 1 only because of insufficient size will be assigned together in stage 2. If FALSE, all objects will be assigned individually.

verbose
: Controls the verbosity of the output. 0 will make the function completely quiet, values up to 4 gradually increase verbosity.

indent
: Controls indentation of printed messages (see verbose above). Each unit adds two spaces before printed messages; useful when several functions' output is to be nested.

Details

The function detects clusters in a hierarchical dendrogram based on the shape of branches on the dendrogram. For details on the method, see http://www.genetics.ucla.edu/labs/horvath/CoexpressionNetwork/BranchCutting.

In order to make the shape parameters `maxCoreScatter` and `minGap` more universal, their values are interpreted relative to `cutHeight` and the 5th percetile of the merging heights (we arbitrarily chose the 5th percetile rather than the minimum for reasons of stability). Thus, the absolute maximum allowable core scatter is calculated as `maxCoreScatter * (cutHeight - refHeight) + refHeight` and the absolute minimum allowable gap as `minGap * (cutHeight - refHeight)`, where `refHeight` is the 5th percentile of the merging heights.

Value

A list containg the following elements:

`labels` Numerical labels of clusters, with 0 meaning unassigned, label 1 labeling the largest cluster etc.

`cores` Numerical labels indicating cores of found clusters.

`smallLabels` Numerical labels for branches that failed to be recognized clusters only because of insufficient number of objects.

`mergeDiagnostics`
A data.frame with one row per merge in the input dendrogram. The columns give the values of the various merging criteria used by the algorithm. Missing data indicate that at least one of the "branches" merged was actually a singleton (single node) and hence the branch merging was automatic.

`mergeCriteria`
Values of the merging thresholds. Either a copy of the corresponding input thresholds or values determined by `deepSplit`.

`branches` A list detailing the deteced branch structure.

Author(s)

Peter Langfelder, <Peter.Langfelder@gmail.com>

References

Langfelder P, Zhang B, Horvath S, 2007. http://www.genetics.ucla.edu/labs/horvath/CoexpressionNetwork/BranchCutting

See Also

`hclust`, `as.dist`

indentSpaces *Spaces for indented output.*

Description

Returns a character string containing two times `indent` spaces.

Usage

```
indentSpaces(indent = 0)
```

Arguments

`indent`	Desired level of indentation. The number of returned spaces will be twice this argument.

Value

A character string containing spaces, of length twice `indent`.

Author(s)

Peter Langfelder, `<Peter.Langfelder@gmail.com>`

Examples

```
spaces = indentSpaces(0);
print(paste(spaces, "This output is not indented..."));
spaces = indentSpaces(1);
print(paste(spaces, "...while this one is."))
```

merge2Clusters *Merge two clusters.*

Description

Merge 2 clusters into 1.

Usage

```
merge2Clusters(labels, mainClusterLabel, minorClusterLabel)
```

Arguments

`labels`	a vector or factor giving the cluster labels
`mainClusterLabel`	label of the first merged cluster. The merged cluster will have this label.
`minorClusterLabel`	label of the second merged cluster.

Value

A vector or factor of the merged labels.

Author(s)

Bin Zhang and Peter Langfelder

Examples

```
options(stringsAsFactors = FALSE);

# Works with character labels:
labels = c(rep("grey", 5), rep("blue", 2), rep("red", 3))
merge2Clusters(labels, "blue", "red")

# Works with factor labels:
labelsF = factor(labels)
merge2Clusters(labelsF, "blue", "red")

# Works also with numeric labels:

labelsN = as.numeric(factor(labels))
labelsN
merge2Clusters(labelsF, 1, 3)
```

---

printFlush *Print arguments and flush the console.*

Description

Passes all its arguments unchaged to the standard `print` function; after the execution of print it flushes the console, if possible.

Usage

```
printFlush(...)
```

Arguments

`...`	Arguments to be passed to the standard `print` function.

Details

Passes all its arguments unchaged to the standard `print` function; after the execution of print it flushes the console, if possible.

Value

Returns the value of the `print` function.

Author(s)

Peter Langfelder, <Peter.Langfelder@gmail.com>

See Also

print

Index