Package 'HCsnip'

April 5, 2014

Type Package

Title Semi-supervised adaptive-height snipping of the Hierarchical Clustering tree

Version 1.2.0

Author Askar Obulkasim

Maintainer Askar Obulkasim <askar703@gmail.com>

Description Decompose given hierarchical clustering tree into non-overlapping clusters in a semi-supervised way by using available patients follow-up information as guidance. Contains functions for snipping HC tree, various cluster quality evaluation criteria, assigning new patients to one of the two given HC trees, testing the significance of clusters with permutation argument and clusters visualization using sample's molecular entropy.

License GPL (>= 2)

LazyLoad yes

Depends
R(>= 2.10.0), survival, coin, fpc, clusterRepro, impute,randomForestSRC, sm, sigaR, Biobase

biocViews Microarray, Bioinformatics, aCGH, GeneExpression, Clustering

R topics documented:

`HCsnip-package` *Semi-supervised adaptive-height snipping of the Hierarchical Clustering tree*

Description

This package contain functions for extracting meaningful clusters from a HC tree. Rather than cutting the HC tree at a fixed highest (as existing methods do), it snips the tree at variable heights to extract hidden clusters. Cluster extraction process uses both the data type from which HC tree is derived and the available patients follow-up information. Functions for testing the significance of extracted clusters and cluster visualization using sample's molecular entropy are also given. If two HC trees are presented which maybe corresponding to the two treatment groups, this package also includes functions for optimally assigning new patients to one of the two HC trees and calculate the expected gain in terms of follow-up.

Details

Package:	HCsnip
Type:	Package
License:	GPL (>= 2)

Author(s)

Askar Obulkasim Maintainer: Askar Obulkasim <askar.wubulikasimu@vumc.nl>

References

Obulkasim,A. et al., (2013). "Semi-supervised adaptive-height snipping of the Hierarchical Clustering tree", submitted.

`BullingerLeukemia` *Leukemia data*

Description

Gene expression profiles of adult acute myeloid leukemia patients. Contains 116 samples and 1571 genes. The follow-up data also included.

Usage

```
data(BullingerLeukemia)
```

Format

An object of list class.

Note

The complete dataset is available at the Gene Expression Omnibus www.ncbi.nlm.nih.gov/geo/, accession number GSE425.

Source

Bullinger,L. et al., (2004). "Use of gene-expression profiling to identify prognostic subclasses in adult acute myeloid leukemia", *N Engl J Med.*, 350, 1605-1616.

Examples

```
data(BullingerLeukemia)
names(BullingerLeukemia)
```

`cluster_pred` *Semi-supervised clustering*

Description

For a given partition, this function assigns new samples to one of the clusters in the partition. Partition (clustering) is composed of non-overlapping clusters

Usage

```
cluster_pred(X, partition, surv.time = NULL, status = NULL, te.index, minclus = 4,
            te.surv.time = NULL, te.status = NULL, method = "conc", maxmiss = 30,
              plot.it = FALSE, ...)
```

Arguments

`X`	An object of class `ExpressionSet` or data matrix in which columns are assumed to represent the samples, and rows represents the sample's features. *X* can also be a square distance matrix or object cass of `dist`. It must include the data set from which *partition* is obtained and the data set of test samples.
`partition`	A numeric vector contains the non-overlapping cluster labels.
`surv.time`	A numeric vector contains follow-up time of patients in the partition.
`status`	A binary vector contains survival status of patients in the partition, 0 = alive, 1 = dead.
`te.index`	A numeric vector contains the indices of columns in *X* corresponds to the test samples.
`minclus`	The minimum number samples allowed to form a cluster for the test set. This is to avoid returning tiny clusters and reduce the effect of outliers.

`te.surv.time`	An optional vector contains follow-up time of patients in the test set. If supplied with *te.status*, the logrank test *p-value* is calculated for the test set.
`te.status`	An optional vector contains survival status of patients in the test set. If supplied with *te.surv.time*, the logrank test *p-value* is calculated for the test set.
`method`	Type of methods to use in assigning test samples to one of the clusters in the partition. Must be either Ward distance 'ward' or Harrel's concordance index 'conc' (default).
`maxmiss`	Maximum percentage of missing values per row in *X*
`plot.it`	If TRUE and follow-up data of the test samples are given, Kaplan Meier curve(s) will be generated for each cluster in the test set.
`...`	Arguments for `impute.knn` from the **impute** package.

Details

User has two options to assign test set to one of the clusters in the partition. One option is to use the Ward distance. Specifically, an average distance is calculated between a test sample and samples in each cluster in the partition, separately. The test sample is assigned to a cluster for which average distance is the smallest. Follow-up data is not required for this option.

Second option is to use the Harrel's concordance index (Harrel et al., 1982). For this option both main and follow-up data corresponds to the given partition are required. Main data is used to find the pseudo nearest neighbours (PNN) of a test sample (Obulkasim et a., 2011), and follow-up data is used to check how much PNN's follow-up info is concordant with follow-up info of samples in each cluster. The test sample is assigned to a cluster for which average concordance is the highest.

Before selecting either one of the options, we recommend user to check the correlation between main data and follow-up info (e.g. using global test). If correlation is relatively large, we recommend to use 'conc' option, and vice versa.

Value

If *plot.it* is FALSE, function returns a vector of predicted cluster labels of the test set. If TRUE and follow-up data of the test set are given, function returns a list object contains following components:

`St` a data frame with following five columns:

- [,1] the unique survival times found in the test set's follow-up.
- [,2] the cluster labels of the test set.
- [,3] the number of patients at risk at each unique time point.
- [,4] the survival probability at each unique time point.
- [,5] the variance of the survival probability at each unique time point.

`value` logrank test *p-value* for the test set

Author(s)

Askar Obulkasim

References

Obulkasim,A. et al., (2013). "Semi-supervised adaptive-height snipping of the Hierarchical Clustering tree", submitted.

Harrel,E.F. et al., (1982). "Evaluating the yield of medical tests", *JAMA*, 247, 2543-2546.

Obulkasim,A. et al., (2011). "Stepwise classification of cancer samples using clinical and molecular data", *BMC Bioinformatics*, 12, 422.

Troyanskaya,O. et al., (2001). "Missing value estimation methods for DNA microarrays". *Bioinformatics*, 17, 520-525.

See Also

TwoHC_assign

Examples

```
data(BullingerLeukemia)
attach(BullingerLeukemia)
cl <- HCsnipper(em[, 1:30], min = 5)
cl <- cl$partitions[cl$id, ]
result <- cluster_pred(X = em[, 1:50], partition = cl[1, ], surv.time = surv.time[1:30],
                       status = status[1:30], te.index = 31:50)
names(result)
```

EnvioPlot *Visualize cluster's molecular entropy by violin plot*

Description

This function first calculates the entopy of each sample in the given partition, and make a violin plot for each cluster. If clusters are different in term of their molecular profiles, then one may expect density differences in the violin plot.

Usage

```
EnvioPlot(X, method = "knn", parti, horizontal = FALSE,
               col = NULL, names = NULL, ...)
```

Arguments

X	An object of class ExpressionSet or data matrix from which partition is obtained. Columns are assumed to represent samples, and rows represent sample's features. Missing values are allowed.
method	Type of method to calculate sample's molecular entropy. Either *knn* or *normal*.
parti	A partition for which violin plot to be made.
horizontal	Should boxes are organized horizontally? default is FALSE.

col	A vector of colors for each cluster. Should be equal the number of clusters in *x*.
names	A vector of labels for each cluster.
...	Arguments for hdEntropy to calculate the molecular entropy from the **sigaR** package.

Value

The entropy estimate is returned as a numeric.

Author(s)

Askar Obulkasim

References

van Wieringen,N.W. and van der Vaart,W.A. (2010) "Statistical analysis of the cancer cell's molecular entropy using high-throughput data", *Bioinformatics*, 27, 556-563.

Adler,D. (2005). vioplot: A violin plot is a combination of a box plot and a kernel density plot, R package, http://cran.r-project.org/web/packages/vioplot/index.html.

See Also

hdEntropy

Examples

```
data(BullingerLeukemia)
attach(BullingerLeukemia)
cl <- cutree(hclust(as.dist(1 - cor(em[, 1:60])), method = "ward"), k = 2)
result <- EnvioPlot(X = em[, 1:60], parti = cl)
```

HCsnipper *HC tree snipper*

Description

This function snips given hierarchical clustering (HC) at variable heights to extract all possible partitions. Each partition (clustering) is composed of non-overlapping clusters.

Usage

```
HCsnipper(X, hc = NULL, dis = NULL, dis.method = "cor", link.method = "ward",
          minclus = 4, maxmiss = 30, ...)
```

Arguments

X	An object of class ExpressionSet or data matrix from which HC tree will be derived. Columns are assumed to represent the samples, and rows represent the sample's features (genes). Missing values are allowed.
hc	HC tree from which partitions to be extracted. Must be an object class of hclust. This is an optional argument, but if given *X* and *dis* will be ignored.
dis	A square distance matrix or object class of dist from which HC tree to be derived. This is an optional argument, if given *X* will be ignored.
dis.method	The distance measure to be used. This must be one of the methods acceptable for dist function or the Pearson correlation 'cor' (default).
link.method	The agglomeration method to be used. This should be one of "ward" (default), "single", "complete", "average", "mcquitty", "median" or "centroid".
minclus	The minimum number of samples allowed to form a cluster. This parameter is inversely proportional to the number of partitions returned. e.g. large values returns less number clusters, and vice versa.
maxmiss	Maximum percentage of missing values per row in *X*
...	Arguments for impute.knn from the impute package for missing values imputation in *X*.

Details

For given HC tree, this function snips it at all possible places to extract partitions under the following conditions:

- Singleton is not allowed.
- Snipping places are chosen so that only the samples which are neighbours in the leaf node ordering (see order(hc)) are allowed to form a cluster.

The last constraint guarantees that sniping does not change the HC tree structure considerably. For example, samples located in far left in the HC tree will not be joined with samples located in far right. The number of partitions return by function depends not only on the *minclus* argument, but also the shape of the HC tree. Large number of partitions can be returned from a balanced HC tree than a skewed one.

Value

This function returns an object of list class contains following objects:

partitions	a matrix in which rows represent partitions and columns represent samples.
id	indices of the partitions in which minimum cluster size is equal or larger than *minclus*.
hc	HC tree from which partitions are extracted.
dat	data matrix. If *X* has missing values, this will be missing values imputed full data matrix.
dis	the distance matrix used
dis.m	the distance measure used
link.m	the agglomeration method used

Author(s)

Askar Obulkasim

References

Obulkasim,A. et al., (2013). "Semi-supervised adaptive-height snipping of the Hierarchical Clustering tree", submitted.

Troyanskaya,O. et al., (2001). "Missing value estimation methods for DNA microarrays". *Bioinformatics*, 17, 520-525.

Examples

```
data(BullingerLeukemia)
attach(BullingerLeukemia)
H <- hclust(as.dist(1 - cor(em[, 1:30])), method = "ward")
cl <- HCsnipper(em[, 1:30], minclus = 5)
cl <- cl$partitions[cl$id, ][1, ]
## Visualize a partition, for this package WGCNA is needed.
#library(WGCNA)
#plotDendroAndColors(H, cl, hang = -1, dendroLabels = FALSE)
```

`measure` *Evaluate cluster quality*

Description

Function to evaluate the overall quality of a parition (composed of non-overlapping clusters) by user defined criteria.

Usage

```
measure(parti, dis, X = NULL, method = "g2", maxmiss = 30, ...)
```

Arguments

`parti`	Partition to be evaluated.
`dis`	A square distance matrix or class object of `dist` corresponding to *x*.
`X`	data matrix corresponding to the *parti*. Columns are assumed to represent the samples, and rows represent the sample's features. Missing values are allowed. This is an optional argument, but If type is set to 'igp', then matrix must be given.
`method`	Type of evaluation measure to use for assessing the quality of clusters in *x*. Default is Goodman and Kruskal index *g2*.
`maxmiss`	Maximum percentage of missing values per row in *dat*
`...`	Arguments for function `cluster.stats` from the **fpc** package. See details below.

Details

Numerous cluster quality measuring criteria have been proposed. This package includes only a few well known ones. Except for the 'c.index' and the in group proportion 'igp', rest of the criteria come from the function `cluster.stats` in **fpc** package. For latter one, please see the returned arguments of the `cluster.stats` function before you decide which criteria to choose. Note that, the value returned by different criteria has different meaning. For example. the larger the Goodman and Kruskal index 'g2' the better, for the index G3 'g3' the smaller the better. Thus, interpret returned value accordingly.

Value

A numeric value representing the quality of partition under consideration.

Author(s)

Askar Obulkasim

References

Hennig,C. (2010). fpc: Flexible procedures for clustering, R package, http://CRAN.R-project.org/package=fpc.

Kapp,A.V. and Tibshirani,R. (2007) "Are clusters found in one dataset present in another dataset?", *Biostatistics*, 8, 9-31.

Obulkasim,A. et al., (2013). "Semi-supervised adaptive-height snipping of the Hierarchical Clustering tree", submitted.

See Also

`surv_measure`

Examples

```
data(BullingerLeukemia)
attach(BullingerLeukemia)
cl <- HCsnipper(em[, 1:30], minclus = 5)
cl <- cl$partitions[cl$id, ]
m <- apply(cl, 1, function(x) measure(parti = x, dis = 1 - cor(em[, 1:30])))
```

`perm_test` *A function to select an optimal partition (clustering) from large number of candidates and calculate the* p*-value for it.*

Description

For a given set of partitions (each partition is composed of non-overlapping clusters), this function uses two types of data to evaluate each partition and select the optimal one which has the highest rank in terms of both data type (presumed that *score1* and *score2* were from two different data source). Permutation approach used to calculate the corrected *p-value* of the selected partition.

Usage

```
perm_test(partitions, surv.time, status, score1 = NULL, score2, method = "BIC", nperm = 1000)
```

Arguments

`partitions`	A matrix in which rows represent partitions and columns represent samples
`surv.time`	A numeric vector contains follow-up time of patients in the partition
`status`	A binary vector contains survival status of patients in the partition, 0 = alive, 1 = dead
`score1`	A numeric vector contains the quality score for each partition. Scores are assumed to be calculated using the follow-up data. Note, prepare this vector in a way that high value corresponds to good quality partition.
`score2`	A numeric vector contains the quality score for each partition calculated by using any data type except for follow-up. The same as *score1* this vector must be prepared in a way that high value corresponds to good quality partition.
`method`	Type of partition evaluation measure to use. Must be the same as the type of measure used in calculating the *score1*. Default is 'BIC'
`nperm`	The number of permutations.

Details

When studying association of cluster membership with follow-up data, we cannot use the standard testing procedures. Because *score1* is already used the follow-up data. Thus, we would use the follow-up data twice and the resulting *p*-value is likely to be too small. We avoid this bias by also applying the semi-supervised partition selection under the null-hypothesis. This null-hypothesis is simply the absence of association between the data type used to generate the *score2* and the follow-up. Our partition selection in combination with a suitable test statistic is designed to detect associations that can be represented by groups of samples. We adapt the p-value computation as follows:

1. Use a suitable test statistic (e.g. log-rank for time-to-event data and chi-square for nominal data) to compute the conditional p-value given the cluster labels in the selected partition: *p_obs*.
2. For i = 1...nperm:
   (a) Randomly permute follow-up data among the samples.
   (b) Apply exactly the same type of evaluation measure to evaluate all partitions, e.g. generate new *score1* , but *score2* is fixed. Selected the best partition as before.
   (c) Conditional on the resulting partition, compute *p*-value *p_i*.

3. Finally, the *p*-value of interest is equal the number of time *p_i* smaller (or equal) than the *p_obs* divided by the number of permutations ran.

Here, *p* satisfies a crucial property of *p*-value: it is uniformly distributed when the null-hypothesis is true, because then *p_obs* and *p_i* are exchangeable random variables. The exchangeability is a result from the null-hypothesis and the use of exactly the same procedures to compute *p_obs* and *p_i*.

Value

A list object contains following objects:

`obs.p`	Observed *p*-value
`perm.p`	A vector of *p*-values from permutations.
`best`	Selected optimal partition

Author(s)

Askar Obulkasim

References

Obulkasim,A. et al., (2013). "Semi-supervised adaptive-height snipping of the Hierarchical Clustering tree", submitted.

See Also

`TwoHC_perm`

Examples

```
data(BullingerLeukemia)
attach(BullingerLeukemia)
cl <- HCsnipper(em[, 1:30], min = 5)
cl <- cl$partitions[cl$id, ]
m <- apply(cl, 1, function(x) measure(parti = x, dis = 1-cor(em[, 1:30])))
s <- apply(cl, 1, function(x) surv_measure(x, surv.time[1:30], status[1:30]))
result <- perm_test(cl, surv.time[1:30], status[1:30], score1 = s, score2 = m, nperm = 10)

### Visualize cluster differences in terms of Entropy.
H <- EnvioPlot(X = em[, 1:30], parti = result$best)
```

RSF_eval *Function to calculate error rate using the Random Survival Forest*

Description

This function constructs survival forest using training sample's follow-up as response and cluster labels as covariate. Constructed forest is used to calculate cumulative hazard function (CHF) for each new sample based its cluster label. CHFs are compared with new samples' actual survival time to calculate the error rate. Error rate ranges from 0 to 1, with 0 representing perfect.

Usage

```
RSF_eval(partition, surv.time, status, te.partition, te.surv.time, te.status, ...)
```

Arguments

partition	Partition (clustering) corresponds to the training set samples. Cluster labels in the partition will be used as covariate to construct SF.
surv.time	A numeric vector contains the follow-up information of patients in *partition* , will be used as response to construct SF.
status	A binary vector contains survival status of patients in the partition, normally 0=alive, 1=dead, will be used as response to construct SF.
te.partition	Partition to be evaluated
te.surv.time	A numeric vector contains the follow-up information of patient's in *te.partition*
te.status	A binary vector contains survival status of patients in *te.partition*, normally 0=alive, 1=dead.
...	Arguments for rfsrc from the **randomForestSRC** package

Value

A vector of error rates. Length is equal the number of trees constructed. Default is 1000.

Author(s)

Askar Obulkasim

References

Ishwaran,H. et al., (2008). "Random survival forest", *Ann. App. Statist.*, 2, 841-860.

Obulkasim,A. et al., (2013). "Semi-supervised adaptive-height snipping of the Hierarchical Clustering tree", submitted.

See Also

rfsrc

Examples

```
data(BullingerLeukemia)
attach(BullingerLeukemia)
cl <- HCsnipper(em[, 1:30], min = 5)
cl <- cl$partitions[cl$id, ]
pred <- cluster_pred(em[, 1:100], cl[1, ], surv.time[1:30],
                     status[1:30], 31:100)
Err <- RSF_eval(cl[1, ], surv.time[1:30], status[1:30], pred,
               surv.time[31:100], status[31:100])
```

---

surv_measure *Cluster quality evaluation using follow-up data*

Description

Function to evaluate the overall quality of a given partition using follow-up data. A partition (clustering) is composed of non-overlapping clusters.

Usage

```
surv_measure(parti, surv.time, status, method = "BIC")
```

Arguments

parti	A partition to be evaluated.
surv.time	A numeric vector contains follow-up time of patient's in x
status	A binary vector contains survival status of patients in x, normally 0=alive, 1=dead.
method	Type of partition evaluation measures to use for assessing the relationship between follow-up and a partition. Default is *BIC*.

Details

This function fits a Cox model using follow-up data as response and cluster labels in the partition as covariate. The likelihood from the fitted model further used to calculate the modified *AIC* or *BIC*. See references for more details. Note that, for convenience in later usage, returned value is multiplied by -1 inside the function so that large value denotes good quality partition.

Value

A numeric value representing the quality of partition under consideration in terms of follow-up.

Author(s)

Askar Obulkasim

References

Liang,H. and Zou,G.H. (2008). "Improved AIC selection strategy for survival analysis", *Comput Stat Anal*., 52, 2538-2548.

Volinsky,T.C. and Raftery,A.E. (2000). "Baysian information criteria for censored survival models", *Biometrics*, 56, 256-262.

Obulkasim,A. et al., (2013). "Semi-supervised adaptive-height snipping of the Hierarchical Clustering tree", submitted.

See Also

measure

Examples

```
data(BullingerLeukemia)
attach(BullingerLeukemia)
cl <- HCsnipper(em[, 1:30], min = 5)
cl <- cl$partitions[cl$id, ]
result <- apply(cl, 1, function(x) surv_measure(x, surv.time[1:30], status[1:30]))
```

TcgaGBM *Glioblastoma multiforme gene expression data*

Description

The subset of latest version of TCGA glioblastoma (GBM) level 3 gene expression data with partial clinical info. Contains expression profiles of 120 samples measured on 3000 genes. Clinical data includes follow-up and type of drugs patients have been administered.

Usage

```
data(TcgaGBM)
```

Format

An object of list class

Note

The complete dataset is available at the TCGA data portal https://tcga-data.nci.nih.gov/tcga/.

Source

The Cancer Genome Atlas Network (2008), "Comprehensive genomic characterisation defines human glioblastoma genes and core pathways", *Nature*, 490, 61-70.

Examples

```
data(TcgaGBM)
names(TcgaGBM)
```

`TwoHC_assign` *Function to assign new samples to one of the two given hierarchical clustering trees in a semi-supervised way*

Description

For given molecular data sets from two non-overlapping groups of patients, this functions constructs two independent HC trees and assigns new samples to one of them in semi-supervised way. See details.

Usage

```
TwoHC_assign(X, index1, index2, new.X, dis.method = "cor", link.method = "ward",
             minclus = 4, maxmiss = 30, surv.time, status, method1 = "BIC",
              method2 = "g2")
```

Arguments

`X`	An object of class `ExpressionSet` or data matrix from which two HC tress to be derived. Columns are assumed to represent the samples, and rows represent the sample's features. Missing values are allowed.
`index1`	Column indices of patients in *X* correspond to the first group.
`index2`	Column indices of patients in *X* correspond to the second group.
`new.X`	An object of class `ExpressionSet` or data matrix corresponds to new samples. Columns are assumed to represent the samples, and rows represents the sample's features. Missing values are allowed.
`dis.method`	The distance measure to be used. This must be one of method acceptable for `dist` function or the Pearson correlation (default).
`link.method`	The agglomeration method to be used. This should be one of "ward" (default), "single", "complete", "average", "mcquitty", "median" or "centroid".
`minclus`	The minimum number of samples allowed to form a cluster. This parameter inversely proportional to the number of partition returned from a HC tree. e.g. a large value returns small number of partitions, and vice versa.
`maxmiss`	Maximum percentage of missing values per row in *X*.
`surv.time`	A numeric vector contains follow-up information of patient's in *X*
`status`	A binary vector contains survival status of patients in *X*, normally 0=alive, 1=dead.
`method1`	Type of partition evaluation measures to use for assessing the relationship between follow-up and a partition. Default is "BIC".
`method2`	Type of Partition evaluation measure to use for assessing the relationship between data matrix *X* and a partition. Default is Goodman and Kruskal index "g2".

Details

Say molecular profiles of two groups patients (without overlap) treated with two different drugs or the same drugs in different combinations are available. Besides that, their follow-up information are also given. When a new patient comes in (for which only molecular profiles are available), question will be to which group this patient should be assigned so that he/she will benefit most by the type of treatment this group received.

This function is designed for this problem. it works as follows: first, two independent HC trees will be derived from given data; second, partitions are extracted and the optimal partition is selected from each HC tree, separately; third, new patient's molecular profile is compared with each cluster in each optimal partition to calculate average similarity and identify two most similar clusters (competing clusters) fromt the two HC trees; finally, new sample is assigned to one of the two competing clusters which has better overall survival.

Value

A list object contains following components:

`hc1`	HC tree derived from the data corresponds to the first group.
`hc2`	HC tree derived from the data corresponds to the second group.
`partitions.hc1`	A matrix includes partitions extracted from *hc1*. Rows represent partitions and columns represent samples.
`partitions.hc2`	A matrix includes partitions extracted from *hc2*. Rows represent partitions and columns represent samples.
`best.hc1`	Optimal partition found on the *hc1*
`best.hc2`	Optimal partition found on the *hc2*
`score.hc1`	A matrix with two columns. The first column contains the quality scores of *partitions.hc1* calculated using the follow-up data. The second column contains the quality scores of *partition.hc1* calculated by using *X*.
`score.hc2`	The same as *score.hc1*, but for *partitions.hc2*.
`Assign`	A matrix with three columns. The first column contains the indices of HC trees to which a test sample was assigned. The second column contains the indices of clusters in *best.hc1* to which a test sample was most similar. The third column contains the indices of clusters in *best.hc2* to which a test sample was most similar.
`surv.time`	The same as input
`status`	The same as input
`index1`	The same as input
`index2`	The same as input
`new.X`	The same as input
`X`	The same as input
`method1`	The same as input
`method2`	The same as input
`minclus`	The same as input

id1	indices of the partitions obtained from the *hc1* in which minimum cluster size is equal or larger than *minclus*.
id2	indices of the partitions obtained from the *hc2* in which minimum cluster size is equal or larger than *minclus*.

Author(s)

Askar Obulkasim

References

Harrel,E.F. et al., (1982). "Evaluating the yield of medical tests", *JAMA*, 247, 2543-2546.

Obulkasim,A. et al., (2011). "Stepwise classification of cancer samples using clinical and molecular data", *BMC Bioinformatics*, 12, 422.

Troyanskaya,O. et al., (2001). "Missing value estimation methods for DNA microarrays". *Bioinformatics*, 17, 520-525.

Obulkasim,A. et al., (2013). "Semi-supervised adaptive-height snipping of the Hierarchical Clustering tree", submitted.

See Also

See also TwoHC_perm, cluster_pred

Examples

```
data(TcgaGBM)
attach(TcgaGBM)
id1 <- which(drugs == "Avastin")
id2 <- which(drugs == "Temodar")
result <- TwoHC_assign(X = em[ ,c(id1[1:30], id2[1:30])], index1 = 1:30, index2 = 31:60,
                       new.X = em[, c(id1[31:60], id2[31:60])], minclus = 4,
                      surv.time = surv.time[c(id1[1:30], id2[1:30])],
                      status = status[c(id1[1:30], id2[1:30])])
```

TwoHC_perm *Function to assess the significance of group assignmetn from TwoHC_assign*

Description

Function to evaluate the significance of the group assignments generated by TwoHC_assign.

Usage

```
TwoHC_perm(TwoHC, nperm = 1000)
```

Arguments

TwoHC	Output from the 'TwoHC_assign' function
nperm	The number of permutations.

Details

Significance of group assignment for each patient is calculated as follows: for a given patient, examine the previously found optimal partition in each HC tree and identify two clusters to which this patient is most similar. Say these two competing clusters are *cluster1* and *cluster2* of size *n1* and *n2*, respectively. Create a binary vector (call it *x*) of size *n1* + *n2* which has *n2* ones and *n1* zeros . Construct a Cox model using follow-up information of samples in *cluster1* and *cluster2* as response, and *x* as covariate. The absolute value of the estimated group parameter ('beta_obs') in the Cox model that compares the survival times of the other patients in the two competing clusters expresses the predicted gain in survival from the assignment by 'TwoHC_assign' with respect to random. The *beta_obs* will be transformed to

$$r^i_{obs} = exp\left(-|\hat{\beta}^i_{obs}|\right),$$

which quantifies the gain in relative risk in the Cox model. The problem is that this is biased, because 'TwoHC_assign' already used 'beta_obs'. Hence, even when the two groups would be equally good for the molecular profiles in the two competing clusters, we obtain 'r_obs' < 1. To correct for this bias this function uses a permutation argument. For each new patient it applies 'nperm' permutations of the surival data among the two competing clusters. As above we compute 'r_perm' for each permutation which contains the same bias as 'r_obs'. Let Z_i = median(r_perm(1), ... , r_perm(nperm)), then risk-ratio rr_obs(i) = r_obs(i) / Z_i quantifies the biased-corrected reduction in relative risk. The permuted version 'rr_perm(i)' is defined analogously (see vignette). Finally it defines a test statistic:

$$T_{obs} = \frac{\frac{1}{n}\sum_{i=1}^{n} log\left(rr^i_{obs}\right)}{stdev\left(log\left(rr^1_{obs}, \ldots, rr^n_{obs}\right)\right)}$$

'T_obs' compared with the background of its null-distribution as obtained by permutation to calculate *p*-value.

Value

A list object contains following objects:

Obs.betas	A numeric vector contains the coefficient from the Cox model corresponding to each test sample.
Perm.betas	A matrix contains the coefficient from the Cox model trained with permuted follow-up data. Columns represent test samples, rows represent permutations
Ranks	A numeric vector contains the rank of each observed coefficient among the *nperm* coefficients generated by permutations.
RiskRatios	A numeric vector contains the ratio of relative risk for the test set.
Pvalue	*p*-value of the overall group assignment.

Author(s)

Askar Obulkasim

References

Obulkasim,A. et al., (2013). "Semi-supervised adaptive-height snipping of the Hierarchical Clustering tree", submitted.

Harrel,E.F. et al., (1982). "Evaluating the yield of medical tests", *JAMA*, 247, 2543-2546.

Obulkasim,A. et al., (2011). "Stepwise classification of cancer samples using clinical and molecular data", *BMC Bioinformatics*, 12, 422.

See Also

See also TwoHC_assign, cluster_pred

Examples

```
data(TcgaGBM)
attach(TcgaGBM)
id1 <- which(drugs == "Avastin")
id2 <- which(drugs == "Temodar")
twoHC <- TwoHC_assign(X = em[ ,c(id1[1:50], id2[1:50])], index1 = 1:50, index2 = 51:100,
                      new.X = em[, c(id1[51:60], id2[51:60])], minclus = 4,
                      surv.time = surv.time[c(id1[1:50], id2[1:50])],
                      status = status[c(id1[1:50], id2[1:50])])
result <- TwoHC_perm(twoHC, nperm = 100)

## Not run:
### Examples with a larger number of permutations (not run).
result <- TwoHC_perm(twoHC, nperm = 10000)
par(mfrow = c(1, 2))
plot(density(result$Ranks), xlab = "Ranks")
plot(density(result$RiskRatios), xlab = "Observed relative risk-ratios")

## End(Not run)
```

Index