Package 'hybridHclust'

July 22, 2015

Title Hybrid Hierarchical Clustering

Version 1.0-5

Date 2015-07-21

Author Hugh Chipman, Rob Tibshirani, with tsvq code originally from Trevor Hastie

Description Hybrid hierarchical clustering via mutual clusters. A mutual cluster is a set of points closer to each other than to all other points. Mutual clusters are used to enrich top-down hierarchical clustering.

Maintainer Hugh Chipman <hugh.chipman@acadiau.ca>

Imports cluster, stats

License GPL-2

NeedsCompilation no

Repository CRAN

Date/Publication 2015-07-22 12:37:37

R topics documented:

`eisenCluster` *An implementation of Eisen's hierarchical clustering*

Description

Bottom-up clustering in which each cluster is represented by the mean vector for observations in the cluster.

Usage

```
eisenCluster(x, method, compatible = TRUE, verbose = FALSE)
```

Arguments

`x`	Data matrix, whose rows we wish to cluster
`method`	How should distance between points (and centres) be calculated? Choices include "euclidean","squared.euclidean", "correlation","uncentered.correlation". For "euclidean" and "squared.euclidean", unexpected behaviour can result, since data points are replaced by their cluster centres, the overall variance in the data will decrease.
`compatible`	Flag for whether cluster merging should be done as in Eisen's cluster algorithm. If `compatible=TRUE`, then when two clusters are merged, a weighted average of the mean vectors for each of the two clusters is used. If `compatible=F`, then the original data are averaged to obtain the new centre. When `x` does not contain missing values, these two options generate the same result. If there are missing values, they will differ. Using the original data makes more sense when there are missing values, since the weights won't account for the missing value pattern.
`verbose`	Prints iteration number if TRUE

Details

The main difference between this algorithm and `hclust(...,method='centroid')` is the manner in which missing values are handled. Here, original rows are merged at each step, taking means after omitting missing observations.

Missing values are permitted, and can be handled in the same manner as in Eisen's package. This is perhaps the main reason the current implementation might be used: to reproduce the clusterings found from Eisen's code when there are missing values. When two clusters are merged, missing values can be handled in two ways (controlled by the `compatible` flag): (1) new cluster centres can be calculated using means of all original observations in the clusters, or (2) new cluster centres can be calculated using a weighted average of the means of the two clusters being joined. Although Eisen's cluster software uses (2), it seems less desirable in situations where observations are missing in some dimensions only, since the presence of missing values will cause the wrong weights to be used when updating centres. Subsequent averaging of clusters centres will ignore the missingness patterns in the cluster means. Option (2) is included to enable clusters identical to Eisen's to be produced.

Value

A `hclust` object. The definition of distance between 2 clusters as the distance between their means can result in a non-monotone dendrogram (e.g., if A, B, C are vertices of an equilateral triangle with side lengths 1, A joins B at distance 1, then C joins AB at distance 0.866). Non-monotone distances are coerced to be monotone before the object is returned. This can yeild dendrograms which seem to join more than 2 points at one height.

The "trueheight" component contains actual heights before they were forced to be monotone.

Author(s)

Hugh Chipman

Examples

```
set.seed(101)
x <- matrix(rnorm(500),5,100)
x <- rbind(x,x[rep(1,4),]+matrix(rnorm(400),4,100))
x <- rbind(x,x[2:5,]+matrix(rnorm(400),4,100))
par(mfrow=c(1,2))
image(1-cor(t(x)),main='correlation distances',zlim=c(0,2),col=gray(1:100/101))
e1 <- eisenCluster(x,'correlation')
plot(e1)
```

---

get.distances *Extract Distances From Mutual Cluster*

Description

Extract the within-cluster distances for each mutual cluster

Usage

```
get.distances(x)
```

Arguments

x	An object of class 'mutualCluster', returned by the 'mutualCluster' function.

Value

A list of distances. For mutual custers of size 2, the corresponding element is a 1x1 matrix. For larger mutual custers, the corresponding element is an object of type 'dist'.

See Also

mutualCluster,print.mutualCluster

Examples

```
x <- cbind(c(-1.4806,1.5772,-0.9567,-0.92,-1.9976,-0.2723,-0.3153),
           c(-0.6283,-0.1065,0.428,-0.7777,-1.2939,-0.7796,0.012))
mc1 <- mutualCluster(x)
get.distances(mc1)
```

`hybridHclust` *Hybrid hierarchical clustering using mutual clusters.*

Description

Top-down clustering (`tsvq`) is applied to data with constraint that mutual clusters cannot be divided. Within each mutual cluster, tsvq is re-applied to yeild a top-down hybrid in which mutual cluster structure is retained.

Usage

```
hybridHclust(x, themc=NULL, trace=FALSE)
```

Arguments

`x`	A data matrix whose rows are to be clustered
`themc`	An object representing the mutual clusters in x, typically generated by `mutualCluster`. If it is not provided, it will be calculated.
`trace`	Should internal steps be printed as they execute?

Details

A mutual cluster is a set of points that should never be broken (see help for 'mutualCluster' for a more precise definition). `hybridHcclust` uses this idea to construct a top-down clustering in which mutual clusters are never broken. This is achieved by temporarily "fusing" together all points in a mutual cluster so that they have equal coordinates, running `tsvq`, and then re-running `tsvq` within each mutual cluster. The resultant top-down clusterings are then "stitched" together to form a single top-down clustering.

Only maximal mutual clusters are constrained to not be broken. Thus if points A, B, C, D are a mutual cluster and points A, B are also a mutual cluster, only the four points will be forbidden from being broken.

Because `hybridHclust` uses `tsvq` to build the hierarchical clusterings, it is implicitly using squared Euclidean distance between rows of x. In some instances (especially for microarray data), a desirable distance measure is d(x1,x2)=1-cor(x1,x2), if x1 and x2 are 2 rows of the matrix x. This correlation-based distance is equivalent to squared Euclidean distance once rows have been scaled to have mean 0 and standard deviation 1. This can be accomplished by pre-processing x before calling `hybridHclust`. An example is provided below.

Value

A dendrogram in `hclust` format.

Author(s)

Hugh Chipman

References

Chipman, H. and Tibshirani, R. (2006) "Hybrid Hierarchical Clustering with Applications to Microarray Data", Biostatistics, 7, 302-317.

See Also

tsvq, "hopach" package

Examples

```
x <- cbind(c(-1.4806,1.5772,-0.9567,-0.92,-1.9976,-0.2723,-0.3153),
c( -0.6283,-0.1065,0.428,-0.7777,-1.2939,-0.7796,0.012))
hyb1 <- hybridHclust(x)
par(mfrow=c(1,2))
plot(x, pch = as.character(1:nrow(x)), asp = 1)
plot(hyb1)

# also works
mc1 <- mutualCluster(x)
mc1
# (3,7) and (1,4) are the two mutual clusters
hyb1 <- hybridHclust(x,mc1)

print('example on sorlie data, may take up to a minute to run')
data(sorlie)
x.scaled <- t(sorlie)
# We take the transpose of "sorlie" because we want to cluster tissue
# samples.  Tissue samples are columns of "sorlie" and hybridHclust will
# cluster rows.

for (i in 1:nrow(x.scaled))
  x.scaled[i,] <- (sorlie[,i]-mean(sorlie[,i]))/sd(sorlie[,i])
# Scale the rows of x.scaled matrix.  This will mean that squared Euclidean
# distance used by hybridHclust will be equivalent to correlation distance.

hhc1 <- hybridHclust(x.scaled,trace=TRUE)
plot(hhc1,labels=dimnames(x.scaled)[[1]])

print('\n\n run demo(hybridHclust) for a more complete package demonstration')
```

---

mutualCluster *Find mutual clusters*

Description

Using bottom-up hierarchical clustering, find the set of maximal mutual clusters.

Usage

```
mutualCluster(x, distances, method = "average", plot = FALSE)
```

Arguments

x	Data matrix, the rows of which we wish to cluster
distances	Distances between objects to be clustered. This may be a symmetric matrix or a object produced by dist. Note that only one of distances and x should be provided.
method	Does not affect mutual clusters returned by mutualCluster. Method used in hclust to join clusters. Must be one of "single", "complete" or "average". This option only affects the plotting, since all 3 methods give the same mutual clusters.
plot	Flag indicating whether the dendrogram for bottom-up clustering should be displayed.

Details

A mutual cluster is a group of points such that the largest distance between any pair in the group is smaller than the shortest distance to any point outside the group.

This function relies on the fact that bottom-up clustering with average, single, or complete linkage cannot break a mutual cluster. That is, when agglomerating, these clustering methods will never add points outside the mutual cluster before first joining all points inside the mutual cluster.

The function mutualCluster is primarily a wrapper that first performs a bottom-up clustering, and then uses this information to identify the mutual clusters. The utility functions that make up mutualCluster are listed under "See Also:" and can be used separately on a hclust object for finer control.

Value

A list of mutual clusters. Each component of the list is a vector of observation indices corresponding to one mutual cluster. Only the maximal mutual clusters are returned, so if observations 1 and 2 form a MC, and observations 1, 2, 4 also form a MC, then a vector with elements 1, 2, 4 will be returned.

Author(s)

Hugh Chipman

References

Chipman, H. and Tibshirani, R. (2006) "Hybrid Hierarchical Clustering with Applications to Microarray Data", Biostatistics, 7, 302-317.

Examples

```
x <- cbind(c(-1.4806,1.5772,-0.9567,-0.92,-1.9976,-0.2723,-0.3153),
           c(-0.6283,-0.1065,0.428,-0.7777,-1.2939,-0.7796,0.012))
par(mfrow=c(1,2))
plot(x,pch=as.character(1:nrow(x)),asp=1) # show data
dist(x) # you can verify that mc's are correct
mutualCluster(x,plot=TRUE) # find MCs and indicate them in dendrogram plot
```

---

print.mutualCluster *Printing Mutual Cluster Objects*

Description

Print method for mutual cluster objects.

Usage

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

Arguments

x	Object of class ‘mutualCluster’
...	Additional arguments to ‘print’ (currently ignored).

See Also

get.distances,mutualCluster

Examples

```
x <- cbind(c(-1.4806,1.5772,-0.9567,-0.92,-1.9976,-0.2723,-0.3153),
           c(-0.6283,-0.1065,0.428,-0.7777,-1.2939,-0.7796,0.012))
mc1 <- mutualCluster(x)
print(mc1)
```

sorlie *Gene expression data for breast cancer tumors*

Description

Gene expression levels of 456 genes for 85 tissue samples. The original data had missing values scattered throughout the matrix. 10-nearest neighbours was used for imputation, as described in Chipman, Hastie and Tibshirani (2003).

Usage

```
data(sorlie)
```

Format

A matrix with 456 rows and 85 columns.

References

Sorlie, T. et. al. (2001) "Gene expression patterns of breast carcinomas distinguish tumor subclasses with clinical implications", PNAS, 98, 10969-74

Chipman, H., Hastie, T, and Tibshirani, R. (2003) 'Clustering Microarray Data' in *Statistical Analysis of Gene Expression Microarray Data*, Terry Speed, Editor, Chapman and Hall, CRC press.

sorlielabels *Cluster labels for sorlie data*

Description

In Sorlie et. al. (2001), M. Eisen's cluster software was used to cluster the 85 samples into 5 groups. These labels contained in the vector sorlie.labels. The naming of the five clusters is as follows:

1: basal-like (14 observations) 2: ERBB2+ (11 observations) 3: Normal (13 observations) 4: Lumnial B/C (15 observations) 5: Luminal A (32 observations)

Usage

```
data(sorlielabels)
```

Format

A vector with 456 elements.

References

Sorlie, T. et. al. (2001) "Gene expression patterns of breast carcinomas distinguish tumor subclasses with clinical implications", PNAS, 98, 10969-74

tsvq *Tree Structured Vector Quantization*

Description

Construct a top-down hierarchical clustering, recursively using k-means with k=2 (kmeans is also known as "vector quantization").

Usage

```
tsvq(x, K=nrow(x), row.labs=1:nrow(x),ntry=20,verbose=FALSE,as.hclust=TRUE,trace=FALSE)
tsvq2hclust(obj)
```

Arguments

x	A data matrix whose rows (i.e., observations) are to be clustered
K	The number of terminal nodes that the tree should have. This must be less than or equal the number of rows of x. If the number of rows of x is used, then the resultant tree will have one data point in each terminal node of the tree.
row.labs	Observation labels. Must be numeric in tsvq for tsvq2hclust to function.
ntry	The number of attempts of 2-means used to subdivide each node into two children. For each attempt, two data points are randomly selected as initial centres. Since 2-means cannot guarantee a globally optimal partition into 2 clusters, multiple tries often will improve the quality of the clustering.
verbose	Should details of the growing be printed?
as.hclust	Should the tree be returned as a hclust object? This option is provided because the hybridHclust function needs the tsvq output in raw form at an intermediate step.
obj	an object created by tsvq
trace	Flag indicating brief iteration count should be printed. Useful for large problems to indicate status.

Details

To construct a top-down hierarchical clustering, the data must be recursively subdivided into two clusters. 2-means is used to find a "good" (but seldom optimal) subdivision. Multiple restarts of kmeans tend to increase the quality of the clustering. Because random seeds are used to select initial centres, two different runs of tsvq are not guaranteed to produce an identical clustering.

The use of k-means implies that the top-down clustering is trying to minimize with-cluster sums of squared Euclidean distances.

Value

If as.hclust=FALSE, then tsvq will return a list that recursively represents the tree structure. If as.hclust=TRUE, an object compatible with hclust is returned. Methods such as plot and cutree can be applied to hclust objects.

The helper function tsvq2hclust will convert a tsvq object to hclust format.

Author(s)

Hugh Chipman; Trevor Hastie wrote the original tsvq code

See Also

hybridHclust

Examples

```
x <- cbind(c(-1.4806,1.5772,-0.9567,-0.92,-1.9976,-0.2723,-0.3153),
c( -0.6283,-0.1065,0.428,-0.7777,-1.2939,-0.7796,0.012))
t1 <- tsvq(x)
par(mfrow=c(1,2))
plot(x,pch=as.character(1:nrow(x)),asp=1)
plot(t1)
cbind(x,cutree(t1,2))
# below also works although you don't need to do it this way.
t2 <- tsvq(x,as.hclust=FALSE)
t2 <- tsvq2hclust(t2)
```

Index