hopach

October 25, 2011

`boot2fuzzy` *function to write MapleTree files for viewing bootstrap estimated*

Description

The MapleTree software (http://mapletree.sourceforge.net/) is an open source, cross-platform, visualization tool to graphically browse results of cluster analyses. The `boot2fuzzy` function takes a data matrix, plus corresponding `hopach` clustering output and bootstrap resampling output, and writes the (.cdt, .fct, and .mb) files needed to view these "fuzzy clustering" results in MapleTree.

Usage

```
boot2fuzzy(data, bootobj, hopach.genes, hopach.arrays = NULL,
file="hopach", clust.wts = NULL, gene.wts = NULL, array.wts = NULL,
gene.names = NULL)
```

Arguments

`data`	data matrix, data frame or exprSet of gene expression measurements. Each column corresponds to an array, and each row corresponds to a gene. All values must be numeric. Missing values are ignored.
`bootobj`	output of `boothopach` or `bootmedoids` applied to the genes - a matrix of bootstrap estimated cluster membership probabilities, with a row for each row in `data` and a column for each cluster.
`hopach.genes`	output of the `hopach` function applied to genes (rows of `data`.
`hopach.arrays`	optional output of the `hopach` function applied to arrays (columns of `data`.
`file`	name for the output files (the extensions .cdt, .mb and .fct will be added).
`clust.wts`	an optional vector of numeric weights for the clusters.
`gene.wts`	an optional vector of numeric weights for the genes.
`array.wts`	an optional vector of numeric weights for the arrays.
`gene.names`	optional vector of names or annotations for the genes, which can be different from the row names of `data`

Value

The function `boot2fuzzy` has no value. It writes three text files to the current working directory.

Note

Thank you to Lisa Simirenko <lsimirenko@lbl.gov> for providing HOPACH views in MapleTree, and to Karen Vranizan <vranizan@uclink.berkeley.edu> for her input. The MapleTree software can be downloaded from: http://sourceforge.net/projects/mapletree/

Author(s)

Katherine S. Pollard <kpollard@gladstone.ucsf.edu>

References

van der Laan, M.J. and Pollard, K.S. A new algorithm for hybrid hierarchical clustering with visualization and the bootstrap. Journal of Statistical Planning and Inference, 2003, 117, pp. 275-303.

http://www.stat.berkeley.edu/~laan/Research/Research_subpages/Papers/hopach.pdf

See Also

hopach, boothopach, bootmedoids, hopach2tree

Examples

```
#25 variables from two groups with 3 observations per variable
mydata<-rbind(cbind(rnorm(10,0,0.5),rnorm(10,0,0.5),rnorm(10,0,0.5)),cbind(rnorm(15,5,0.5
dimnames(mydata)<-list(paste("Var",1:25,sep=""),paste("Exp",1:3,sep=""))
mydist<-distancematrix(mydata,d="cosangle") #compute the distance matrix.

#clusters and final tree
clustresult<-hopach(mydata,dmat=mydist)

#bootstrap resampling
myobj<-boothopach(mydata,clustresult)

#write MapleTree files
boot2fuzzy(mydata,myobj,clustresult)
```

---

bootplot *function to make a barplot of bootstrap estimated cluster membership*

Description

After clustering, the boothopach or bootmedoids function can be used to estimated the membership of each element being clustered in each of the identified clusters (fuzzy clustering). The proportion of bootstrap resampled data sets in which each element is assigned to each cluster is called the "reappearance proportion" for the element and that cluster. This function plots these proportions in a colored barplot.

Usage

```
bootplot(bootobj, hopachobj, ord = "bootp", main = NULL, labels = NULL,
showclusters = TRUE, ...)
```

Arguments

bootobj	output of boothopach or bootmedoids applied to the genes - a matrix of bootstrap estimated cluster membership probabilities, with a row for each row in data and a column for each cluster.
hopachobj	output of the hopach function. If bootobj was generated using bootmedoids (i.e. hopach was not run), then the bootplot function can be used by creating a hopachobj which is a list with at least the following two components: hopachobj$clustering$sizes (number of elements in each cluster - length should be ncol(bootobj) and hopachobj$clustering$order (an ordering of the elements so that elements in the same cluster appear next to each other and elements may also be ordered within cluster). By changing the value of hopachobj$clustering$order, the order of the elements in the barplot can be altered.
ord	character string indicating how to order the elements (rows) in the barplot. If ord="none", then the elements are plotted in the same order as in bootobj, i.e. the same order as the original data matrix. If ord="final", the ordering of elements in the final level of the hopach hierarchical tree is used. If ord="cluster", the ordering from the level of the hopach tree corresponding to the main clusters is used. If ord="bootp", the elements are ordered first by main cluster and then by bootstrap reappearance proportion within cluster, so that elements with the highest membership in the cluster appear at the bottom. In the last three cases, the elements from each cluster will be contiguous. If ord="final", then the medoid element will appear in the middle of each cluster. If ord="clust", the ordering depends on the value of the ord argument passed to the hopach function. For example, when ord="own" in hopach, the elements are ordered within cluster based on distance to the medoid, so that the medoid appears first (at the bottom) in the cluster.
main	character string to be used as the main title
labels	a vector of labels for the elements being clustered to be used on the axes. If the number of elements is lager than 50, the labels are not shown.
showclusters	indicator of whether or not to show the cluster boundaries on the plot. If show.clusters=TRUE, solid lines are drawn at the edges of the clusters.
...	additional arguments to the barplot plotting function

Details

Each cluster (column of bootobj) is represented by a color. The proportion of bootstrap resampled data sets in which an element appeared in that cluster determines the proportion of the bar for that element which is the corresponding color. As a key, the clusters are labeled on the right margin in text of the same color.

Value

The function bootplot has no value. It does generate a plot.

Note

Thank you to Sandrine Dudoit <sandrine@stat.berkeley.edu> for her input and to Jenny Bryan for the original clusplot code.

Author(s)

Katherine S. Pollard <kpollard@gladstone.ucsf.edu>

References

van der Laan, M.J. and Pollard, K.S. A new algorithm for hybrid hierarchical clustering with visualization and the bootstrap. Journal of Statistical Planning and Inference, 2003, 117, pp. 275-303.

http://www.stat.berkeley.edu/~laan/Research/Research_subpages/Papers/hopach.pdf

See Also

hopach, boothopach, bootmedoids, barplot

Examples

```
mydata<-rbind(cbind(rnorm(10,0,0.5),rnorm(10,0,0.5),rnorm(10,0,0.5)),cbind(rnorm(15,5,0.5
dimnames(mydata)<-list(paste("Var",1:25,sep=""),paste("Exp",1:3,sep=""))
mydist<-distancematrix(mydata,d="euclid")

#hopach clustering
clustresult<-hopach(mydata,dmat=mydist)

#bootstrap
myobj<-boothopach(mydata,clustresult)

#plots
bootplot(myobj,clustresult,showclusters=FALSE)
bootplot(myobj,clustresult,labels=paste("Sample",LETTERS[1:25],sep=" "))
```

---

boothopach *functions to perform non-parametric bootstrap resampling of hopach*

Description

The function boothopach takes gene expression data and corresponding hopach gene clustering output and performs non-parametric bootstrap resampling. The medoid genes (cluster profiles) from the original hopach clustering result are fixed, and in each bootstrap resampled data set, each gene is assigned to the closest medoid. The proportion of bootstrap samples in which each gene appears in each cluster is an estimate of the gene's membership in each cluster. These membership probabilities can be viewed as a "fuzzy" clustering result. The function bootmedoids take medoids and a distance function, rather than a hopach object, as input.

Usage

```
boothopach(data, hopachobj, B = 1000, I, hopachlabels = FALSE)

bootmedoids(data, medoids, d = "cosangle", B = 1000, I)
```

Arguments

data	data matrix, data frame or exprSet of gene expression measurements. Each column corresponds to an array, and each row corresponds to a gene. All values must be numeric. Missing values are ignored.
hopachobj	output of the hopach function.
B	number of bootstrap resampled data sets.
I	number of bootstrap resampled data sets (deprecated, retaining til v1.2 for back compatibility).
hopachlabels	indicator of whether to use the hopach cluster labels hopachobj$clustering$labels for the row names (TRUE) versus the numbers 0 to 'k-1', where 'k' is the number of clusters (FALSE).
medoids	row indices of data for the cluster medoids.
d	character string specifying the metric to be used for calculating dissimilarities between vectors. The currently available options are "cosangle" (cosine angle or uncentered correlation distance), "abscosangle" (absolute cosine angle or absolute uncentered correlation distance), "euclid" (Euclidean distance), "abseuclid" (absolute Euclidean distance), "cor" (correlation distance), and "abscor" (absolute correlation distance). Advanced users can write their own distance functions and add these.

Details

The function boothopach requires only data and the corresponding output from the HOPACH clustering algorithm produced by the hopach function. The function bootmedoids is designed to work for any clustering result; the user imputs data, medoid row indices, and the distance metric. The supplied distance metrics are the same as for the distancematrix function. Each non-parametric bootstrap resampled data set consists of resampling the 'n' columns of data with replacement 'n' times. The distance between each element and each of the medoid elements is computed using d for each bootstrap data set, and every element is assigned (for that resampled data set) to the cluster whose medoid is closest. These bootstrap cluster assignments are tabulated over all I bootstrap data sets.

Value

A matrix of bootstrap estimated cluster membership probabilities, which sum to 1 (over the clusters) for each element being clustered. This matrix has one row for each element being clustered and one column for each of the original clusters (one cluster for each medoid). The value in row 'j' and column 'i' is the proportion of the I bootstrap resampled data sets that element 'j' appeared in cluster 'i' (i.e. was closest to medoid 'i').

Author(s)

Katherine S. Pollard <kpollard@gladstone.ucsf.edu> and Mark J. van der Laan <laan@stat.berkeley.edu>

References

van der Laan, M.J. and Pollard, K.S. A new algorithm for hybrid hierarchical clustering with visualization and the bootstrap. Journal of Statistical Planning and Inference, 2003, 117, pp. 275-303.

http://www.stat.berkeley.edu/~laan/Research/Research_subpages/Papers/hopach.pdf

http://www.bepress.com/ucbbiostat/paper107/

http://www.stat.berkeley.edu/~laan/Research/Research_subpages/Papers/jsmpaper.pdf

Kaufman, L. and Rousseeuw, P.J. (1990). Finding Groups in Data: An Introduction to Cluster Analysis. Wiley, New York.

See Also

distancematrix, hopach

Examples

```
#25 variables from two groups with 3 observations per variable
mydata<-rbind(cbind(rnorm(10,0,0.5),rnorm(10,0,0.5),rnorm(10,0,0.5)),cbind(rnorm(15,5,0.5
dimnames(mydata)<-list(paste("Var",1:25,sep=""),paste("Exp",1:3,sep=""))
mydist<-distancematrix(mydata,d="cosangle") #compute the distance matrix.

#clusters and final tree
clustresult<-hopach(mydata,dmat=mydist)

#bootstrap resampling
myobj<-boothopach(mydata,clustresult)
table(apply(myobj,1,sum)) # all 1
myobj[clustresult$clust$medoids,] # identity matrix
```

---

correlationordering

function to compute empirical correlation between distance in a list

---

Description

Given a matrix of pair wise distances based on a choice of distance metric, correlationordering computes the empirical correlation (over all pairs of elements) between the distance apart in the rows/columns of the matrix and the distance according to the metric. Correlation ordering will be high if elements close to each other in the matrix have small pair wise distances. If the rows/columns of the distance matrix are ordered according to a clustering of the elements, then correlation ordering should be large compared to a matrix with randomly ordered rows/columns.

Usage

```
correlationordering(dist)

improveordering(dist,echo=FALSE)
```

Arguments

dist	matrix of all pair wise distances between a set of 'p' elements, as produced, for example, by the distancematrix function. The value in row 'j' and column 'i' is the distance between element 'i' and element 'j'. The matrix must be symmetric. The ordering of the rows/ columns is compared to the values in the matrix.

echo	indicator of whether the value of correlation ordering before and after rearranging the ordering should be printed.

Details

Correlation ordering is defined as the empirical correlation between distance in a list and distance according to some other metric. The value in row 'i' and column 'j' of dist is compared to 'j-i'. The function correlationordering computes the correlation ordering for a matrix dist, whereas the function improveordering swaps the ordering of elements in dist until doing so no longer improves correlation ordering. The algorithm for improveordering is not optimized, so that the function can be quite slow for more than 50 elements. These functions are used by the hopach clustering function to sensibly order the clusters in the first level of the hierarchical tree, and can also be used to order elements within clusters when the number of elements is not too large.

Value

For correlationordering, a number between -1 and 1, as returned by the cor function, equal to the correlation ordering for the matrix dist.

For improveordering, a vector of length 'p' containing the row indices for the new ordering of the rows/columns of dist, so that dist[improveordering(dist)] now has higher correlation ordering.

Warning

The function improveordering can be very slow for more than about 50 elements. The method employed is a greedy, step-wise algorithm, in which sequentially swaps all pairs of elements and accepts any swap that improves correlation ordering.

Author(s)

Katherine S. Pollard <kpollard@gladstone.ucsf.edu> and Mark J. van der Laan <laan@stat.berkeley.edu>

References

van der Laan, M.J. and Pollard, K.S. A new algorithm for hybrid hierarchical clustering with visualization and the bootstrap. Journal of Statistical Planning and Inference, 2003, 117, pp. 275-303.

http://www.stat.berkeley.edu/~laan/Research/Research_subpages/Papers/hopach.pdf

See Also

distancematrix, hopach

Examples

```
mydata<-matrix(rnorm(50),nrow=10)
mydist<-distancematrix(mydata,d="euclid")
image(as.matrix(mydist))
correlationordering(mydist)
neword<-improveordering(mydist,echo=TRUE)
correlationordering(mydist[neword,neword])
image(as.matrix(mydist[neword,neword]))
```

`distancematrix` *functions to compute pair wise distances between vectors*

Description

The function `distancematrix` is applied to a matrix of data to compute the pair wise distances between all rows of the matrix. In hopach versions >= 2.0.0 these distance functions are calculated in C, rather than R, to improve run time performance. function `distancevector` is applied to a matrix and a vector to compute the pair wise distances between each row of the matrix and the vector. Both functions allow different choices of distance metric. The functions `dissmatrix` and `dissvector` allow one to convert between a distance matrix and a vector of the upper triangle. The function `vectmatrix` is used internally.

Usage

```
distancematrix(X, d, na.rm=TRUE)

distancevector(X, y, d, na.rm=TRUE)

dissmatrix(v)

dissvector(M)

vectmatrix(index, p)
```

Arguments

`X`	a numeric matrix. Missing values will be ignored if na.rm=TRUE.
`y`	a numeric vector, possibly a row of X. Missing values will be ignoredif na.rm=TRUE.
`na.rm`	an indicator of whether or not to remove missing values. If na.rm=TRUE (default), then distances are computed over all pairwise non-missing values. Else missing values are propagated through the distance computation.
`d`	character string specifying the metric to be used for calculating dissimilarities between vectors. The currently available options are "cosangle" (cosine angle or uncentered correlation distance), "abscosangle" (absolute cosine angle or absolute uncentered correlation distance), "euclid" (Euclidean distance), "abseuclid" (absolute Euclidean distance), "cor" (correlation distance), and "abscor" (absolute correlation distance). Advanced users can write their own distance functions and add these.
`M`	a symmetric matrix of pair wise distances.
`v`	a vector of pair wise distances corresponding to the upper triangle of a distance matrix, stored by rows.
`index`	index in a distance vector, like that returned by `dissvector`.
`p`	number of elements, e.g. the number of rows in a distance matrix.

Details

In hopach versions <2.0.0, these functions returned the square root of the usual distance for `d="cosangle"`, `d="abscosangle"`, `d="cor"`, and `d="abscor"`. Typically, this transformation makes the dissimilarity correspond more closely with the norm. In order to agree with the `dist` function, the square root is no longer used in versions >=2.0.0.

Value

For versions >= 2.0.0 `distancematrix`, a `hdist` object of of all pair wise distances between the rows of the data matrix 'X', i.e. the value of `hdist[i,j]` is the distance between rows 'i' and 'j' of 'X', as defined by 'd'. A `hdist` object is an S4 class containing four slots:

`Data`	representing the lower triangle of the symmetric distance matrix.
`Size`	the number of objects (i.e. rows of the data matrix).
`Labels`	labels for the objects, usually the numbers 1 to Size.
`Call`	the distance used in the call to `distancematrix`.

A hdist object and can be converted to a matrix using `as.matrix(hdist)`. (See `hdist` for more details.)

For `distancevector`, a vector of all pair wise distances between rows of 'X' and the vector 'y'. Entry 'j' is the distance between row 'j' of 'X' and the vector 'y'.

For `distancevector`, a vector of all pair wise distances between rows of 'X' and the vector 'y'. Entry 'j' is the distance between row 'j' of 'X' and the vector 'y'.

For `dissmatrix`, the corresponding distance vector. For `dissvector`, the corresponding distance matrix. If 'M' has 'p' rows (and columns), then 'v' is length 'p*(p-1)/2'.

For `vectmatrix`, the indices of the row and column of a distance matrix corresponding to entry `index` in the corresponding distance vector.

Warning

The correlation and absolute correlation distance functions call the `cor` function, and will therefore fail if there are missing values in the data and na.rm!=TRUE.

Author(s)

Katherine S. Pollard <kpollard@gladstone.ucsf.edu> and Mark J. van der Laan <laan@stat.berkeley.edu>, with Greg Walll

References

van der Laan, M.J. and Pollard, K.S. A new algorithm for hybrid hierarchical clustering with visualization and the bootstrap. Journal of Statistical Planning and Inference, 2003, 117, pp. 275-303.

http://www.stat.berkeley.edu/~laan/Research/Research_subpages/Papers/hopach.pdf

See Also

`hopach`, `correlationordering`, `disscosangle`

Examples

```
mydata<-matrix(rnorm(50),nrow=10)
deuclid<-distancematrix(mydata,d="euclid")
# old method vdeuclid<-dissvector(deuclid)
vdeuclid<-deuclid@Data
ddaisy<-daisy(mydata)
vdeuclid
ddaisy/sqrt(length(mydata[1,]))

d1<-distancematrix(mydata,d="abscosangle")
d2<-distancevector(mydata,mydata[1,],d="abscosangle")
d1[1,]
d2 #equal to d1[1,]

# old method d3<-dissvector(d1)
d3<-d1@Data
pair<-vectmatrix(5,10)
d1[pair[1],pair[2]]
d3[5]
```

`dplot` *function to make a pseudo-color image of a distance matrix with the row*

Description

The `hopach` clustering function orders the elements being clustered. This ordering can be used to rearrange the rows and columns in the corresponding distance matrix. A pseudo-color image of the ordered distance matrix will reveal the underlying patterns in the clustered data.

The functions 'heat.colors', 'terrain.colors' and 'topo.colors' create heat-spectrum (red to white) and topographical color schemes suitable for displaying ordered data, with 'n' giving the number of colors desired.

Usage

```
dplot(dist, hopachobj, ord = "final", col = heat.colors(12), main = NULL,
xlab = NULL, ylab = NULL, labels = NULL, showclusters = TRUE, ...)
```

Arguments

`dist`	matrix of all pair wise distances between a set of 'p' elements, as produced, for example, by the `distancematrix` function. The value in row 'j' and column 'i' is the distance between element 'i' and element 'j'. The matrix must be symmetric. The distance metric should be the same as that used in the `hopach` function.
`hopachobj`	output of the `hopach` function.
`ord`	character string indicating which of the two orderings produced by `hopach` should be used for the plot. If ord="final", the ordering of elements in the final level of the hierarchical tree is used. If ord="cluster", the ordering from the level of the tree corresponding to the main clusters is used. In both cases, the elements from each cluster will be contiguous. If ord="final", then the medoid element

	will appear in the middle of each cluster. Else, the ordering depends on the value of `ord` passed to the `hopach` function. If ord="none", then the elements are plotted in the same order as in dist.
`col`	a list of colors such as that generated by 'rainbow', 'heat.colors', 'topo.colors', 'terrain.colors' or similar functions.
`main`	character string to be used as the main title
`xlab`	character string to be used as the horizontal axis label. If NULL, the label will be "" (no label).
`ylab`	character string to be used as the vertical axis label. If NULL, the label will be "" (no label).
`labels`	a vector of labels for the elements being clustered to be used on the axes. If labels=NULL, no axes are plotted - this is useful when there are a large number of elements being plotted.
`showclusters`	indicator of whether or not to show the cluster boundaries on the plot. If show.clusters=TRUE, dotted lines are drawn at the edges of the clusters.
`...`	additional arguments to the `image` plotting function

Note

Thank you to Sandrine Dudoit <sandrine@stat.berkeley.edu> for her input.

Author(s)

Katherine S. Pollard <kpollard@gladstone.ucsf.edu> and Mark J. van der Laan <laan@stat.berkeley.edu>

References

van der Laan, M.J. and Pollard, K.S. A new algorithm for hybrid hierarchical clustering with visualization and the bootstrap. Journal of Statistical Planning and Inference, 2003, 117, pp. 275-303.

http://www.stat.berkeley.edu/~laan/Research/Research_subpages/Papers/hopach.pdf

See Also

`distancematrix`, `hopach`, `image`

Examples

```
mydata<-matrix(rnorm(50),nrow=10)
mydist<-distancematrix(mydata,d="euclid")
clustresult<-hopach(mydata,dmat=mydist)
dplot(mydist,clustresult,showclusters=FALSE)
dplot(mydist,clustresult,col=topo.colors(15))
```

golub — *Gene expression dataset from Golub et al. (1999)*

Description

Gene expression data (3051 genes and 38 tumor mRNA samples) from the leukemia microarray study of Golub et al. (1999). Pre-processing was done as described in Dudoit et al. (2002). The R code for pre-processing is available in the file golub.R in the docs directory.

Usage

```
data(golub)
```

Value

`golub`	matrix of gene expression levels for the 38 tumor mRNA samples, rows correspond to genes (3051 genes) and columns to mRNA samples.
`golub.cl`	numeric vector indicating the tumor class, 27 acute lymphoblastic leukemia (ALL) cases (code 0) and 11 acute myeloid leukemia (AML) cases (code 1).
`golub.gnames`	a matrix containing the names of the 3051 genes for the expression matrix `golub`. The three columns correspond to the gene `index`, `ID`, and `Name`, respectively.

Source

Golub et al. (1999). Molecular classification of cancer: class discovery and class prediction by gene expression monitoring, *Science*, Vol. 286:531-537.
http://www-genome.wi.mit.edu/MPR/ .

References

S. Dudoit, J. Fridlyand, and T. P. Speed (2002). Comparison of discrimination methods for the classification of tumors using gene expression data. *Journal of the American Statistical Association*, Vol. 97, No. 457, p. 77–87.

hdist-class — *Class "hdist" - S4 class to hold distance matrices.*

Description

Class hdist was created to take advantage of the structure innate to symmetric matrices. It stores only the lower triangle of the matrix, thus reducing the size (and memory usage) from n x n to [n x (n - 1)] / 2.

Like a matrix, a hdist object is subsettable; thus, hdist[i,j] will return the value at row 'i' column 'j'. Most valid indices for a matrix are also valid for a hdist object. (See examples below)

Slots

`Data`: Object of class `"numeric"` a vector containing the stacked columns of the lower triangle of a symmetric matrix – often the symmetric matrix is a distance matrix.

`Size`: Object of class `"numeric"` the dimension of the symmetric matrix, from which `Data` was constructed.

`Labels`: Object of class `"numeric"` a list of values of length `Size` to allow for pretty printing.

`Call`: Object of class `"character"` a character string specifying the method used to create the distance matrix from which `Data` was constructed.

Methods

hdist `signature{Data = "numeric", Size = "numeric", Labels = "numeric", Call = "character"}`: Create a new hdist object.

as.hdist `signature{from = "matrix"}`: Converts a matrix to a hdist object.

as.matrix `signature(x = "hdist")`: Converts a hdist object to a matrix.

as.vector `signature(x = "hdist", mode = "missing")`: Returns the hdist object as a vector.

[`signature(x = "hdist")`: Subsetting function for hdist objects. See examples and warning.

coerce `signature(from = "matrix", to = "hdist")`: Converts a matrix to a hdist object.

coerce `signature(from = "hdist", to = "matrix")`: Converts a hdist object to a matrix.

dim `signature(x = "hdist")`: Returns the dimension of the hdist object if expanded to a square matrix.

labels `signature(object = "hdist")`: Returns the labels used for printing.

length `signature(x = "hdist")`: Returns the number of rows in hdist object.

show `signature(object = "hdist")`: Prints the hdist object.

Warning

A hdist object is NOT closed under the subsetting operation. For instance, if a 100 x 100 symmetric matrix is stored as an hdist object, hdist[c(3,4,5),c(7,8,9)] will return a 3 x 3 matrix, since the subsetting will not result in a symmetric matrix. However, if index i = j, then subsetting a hdist object will result in a symmetric matrix, and thus a hdist object will be returned. (See examples below)

Note

Thank you to Larry Tai for his assistance creating run-time comparisons.

Author(s)

Katherine S. Pollard <kpollard@gladstone.ucsf.edu> and Gregory D. Wall <gwall@wald.ucdavis.edu>

References

van der Laan, M.J. and Pollard, K.S. A new algorithm for hybrid hierarchical clustering with visualization and the bootstrap. Journal of Statistical Planning and Inference, 2003, 117, pp. 275-303.

http://www.stat.berkeley.edu/~laan/Research/Research_subpages/Papers/hopach.pdf

http://www.bepress.com/ucbbiostat/paper107/

http://www.stat.berkeley.edu/~laan/Research/Research_subpages/Papers/jsmpaper.pdf

Kaufman, L. and Rousseeuw, P.J. (1990). Finding Groups in Data: An Introduction to Cluster Analysis. Wiley, New York.

See Also

hopach

Examples

```
showClass("hdist")

library(hopach)
X <- matrix(rnorm(60,mean=10,sd=2),nrow=10,ncol=6,byrow=TRUE)
dmat <- disscosangle(X)
dmat
str(dmat)

# Examples where subsetting a hdist object returns a matrix...
dmat[c(3,4,5),c(5,6,7,8)]
dmat[c(TRUE,FALSE),c(FALSE,TRUE)]
dmat[c(4,5,6), ]

# Examples where subsetting a hdist object returns a hdist object...
dmat[c(3,4,5,6,7),c(3,4,5,6,7)]
dmat[c(TRUE,FALSE),c(TRUE,FALSE)]

# Expand hdist object to a symmetric matrix...
as.matrix(dmat)
```

---

hopach *function to perform HOPACH hierarchical clustering*

Description

The Hierarchical Ordered Partitioning and Collapsing Hybrid (HOPACH) clustering algorithm builds a hierarchical tree by recursively partitioning a data set (e.g., gene expression measurements) with the PAM algorithm, while ordering and possibly collapsing clusters at each level. The algorithm uses the Mean/Median Split Silhouette (MSS) criteria to identify the level of the tree with maximally homogeneous clusters. It also runs the tree down to produce a final ordered list of the elements.

Usage

```
hopach(data, dmat = NULL, d = "cosangle", clusters = "best", K = 15,
kmax = 9, khigh = 9, coll = "seq", newmed = "medsil", mss = "med",
impr = 0, initord = "co", ord = "own", verbose=FALSE)
```

Arguments

`data`	data matrix, data frame or exprSet of gene expression measurements. Typically, each column corresponds to an array, and each row corresponds to a gene. For clustering arrays, the arrays appear in the rows and the genes in the columns. All values must be numeric. Missing values are ignored.
`dmat`	matrix or `hdist` object of pair wise distances between all genes (arrays). All values must be numeric, and missing values are not allowed. If NULL, this matrix is computed using the metric specified by `d`. If a matrix is provided, the user is responsible for ensuring that the metric used agrees with `d`.
`d`	character string specifying the metric to be used for calculating dissimilarities between variables. The currently available options are "cosangle" (cosine angle or uncentered correlation distance), "abscosangle" (absolute cosine angle or absolute uncentered correlation distance), "euclid" (Euclidean distance), "abseuclid" (absolute Euclidean distance), "cor" (correlation distance), and "abscor" (absolute correlation distance). Advanced users can write their own distance functions and add these to the functions `distancematrix()` and `distancevector()`.
`clusters`	character string specifying if clusters are to be identified as the level of the tree with the minimum mean/median split silhouette (MSS) ("best"), the first level of the tree below which MSS increases ("greedy"), or not at all ("none").
`K`	positive integer specifying the maximum number of levels in the tree. Must be 15 or less, due to computational limitations (overflow).
`kmax`	integer between 1 and 9 specifying the maximum number of children at each node in the tree.
`khigh`	integer between 1 and 9 specifying the maximum number of children at each node in the tree when computing MSS. Can be different from kmax, though typically these are the same value.
`coll`	character string specifying how collapsing steps are performed at each level. The options are "seq" (begin with the closest pair of clusters and collapse pairs sequentially as long as MSS decreases) and "all" (consider all pairs of clusters and collapse any that decrease MSS).
`newmed`	character string specifying how to choose a medoid for the new cluster after collapsing a pair of clusters. The options are "medsil" (maximizer of medoid based silhouette, i.e.: (a-b)/max(a,b), where a is distance to medoid and b is distance to next closest medoid), "nn" (nearest neighbor of mean of two collapsed cluster medoids weighted by cluster size), "uwnn" (unweighted version of nearest neighbor, i.e. each cluster - rather than each element - gets equal weight), "center" (minimizer of average distance to the medoid).
`mss`	character vector specifying what criteria function to use. The options are "med" (median split silhouette) or "mean" (mean split silhouette). See details for definition of split silhouettes. The MSS criteria is used to determine the number of children at each node, to decide what collapsing should be performed at each level, and to determine the main clusters.

`impr` number between 0 and 1 specifying the margin of improvement in MSS needed to accept a collapse step. If (MSS.before - MSS.after)/MSS.before is less than `impr`, then the collapse is not performed.

`initord` character string specifying how to order the clusters in the initial level of the tree. The options are "co" (maximize correlation ordering, i.e. the empirical correlation between distance apart in the ordering and distance between the cluster medoids) or "clust" (apply hopach with binary splits to the cluster medoids and use the final level of that tree as the ordering). In subsequent levels, the clusters are ordered relative to the previous level, so this initial ordering determines the overall structure of the tree.

`ord` character string specifying how to order the elements within clusters. This method is used to create an ordering of all elements at the level of the tree corresponding to the main clusters. The options are "own" (order based on distance from cluster medoid with medoid first, i.e. leftmost), "neighbor" (order based on distance to the medoid of the next cluster to the right), or "co" (maximize correlation ordering - can be slow for large clusters!).

`verbose` If `TRUE` then verbose output is printed.

Details

The HOPACH hierarchical clustering algorithm is a hybrid between an agglomerative (bottom up) and a divisive (top down) algorithm. The HOPACH tree is built from the root node (all elements) down to the leaf nodes, but at each level collapsing steps are used to unite similar clusters. In addition, the clusters in each level are ordered with a deterministic algorithm based on the same distance metric that is used in the clustering. In this way, the ordering produced in the final level of the tree does not depend on the order of the data in the original data set (as can be the case with algorithms that have a random component in their ordering methods). Unlike other hierarchical clustering methods, HOPACH builds a tree of clusters in which the nodes need not be binary, i.e. there can be more than two children at each split. The divisive steps of the HOPACH algorithm are performed using the PAM algorithm described in chapter 2 of Kaufman and Rousseeuw (1990) and the R package 'cluster'.

The Median (or Mean) Split Silhouette (MSS) criteria is used by HOPACH to (i) determine the optimal number of children at each node, (ii) decide which pairs of clusters to collapse at each level, and (iii) identify the first level of the tree with maximally homogeneous clusters. In each case, the goal is to minimize MSS, which is a measure of cluster heterogeneity described in http://www.bepress.com/ucbbiostat/paper107/

In hopach versions <2.0.0, these functions returned the square root of the usual distance for `d="cosangle"`, `d="abscosangle"`, `d="cor"`, and `d="abscor"`. Typically, this transformation makes the dissimilarity correspond more closely with the norm. In order to agree with the `dist` function, the square root is no longer used in versions >=2.0.0. See ? distancematrix().

Value

A list with the following components:

`clustering` the partitioning or 'main clusters' with the following components

'k' is an integer specifying the number of clusters identified by minimizing MSS.

'medoids' is a vector indicating the rows of `data` that are the 'k' cluster medoids, i.e. profiles (or centroids) for each cluster.

'sizes' is a vector containing the 'k' cluster sizes.

'labels' is a vector containing the main cluster labels for every variable. Each label consists of one digit per level of the tree (up to the level identified as the

main clusters). The digit (1-9) indicates which child cluster the variable was in at that level. For example, '124' means the fist (leftmost in the tree) cluster in level 1, the second child of cluster '1' in level 2, and the fourth child of cluster '12' in level 3. These can be mapped to the numbers 1:k for simplicity, though the tree structure and relationship amongst the clusters is then lost, e.g. 1211 is closer to 1212 than to 1221.

'order' is a vector containing the ordering of variables within the main clusters. The clusters are ordered deterministically as the tree is built. The elements within each of the main clusters are ordered with the method determined by the value of `ord`: "own" (relative to own medoid), "neighbor" (relative to next medoid to the right), or "co" (maximize correlation ordering).

`final` the final level of the hierarchical tree with the following components

'labels' is a vector containing the final labels for every variable. Each label consists of one digit per level of the tree (up to the final level), and the format for the labels is the same as for the clustering labels. The final labels contain the entire history of the tree. In fact, internal level 'n' can be reproduced by truncating the final labels to 'n' digits. Ordering the final labels produces the final ordering (final level of the tree), while ordering internal level labels produces an ordering of the clusters at that level.

'order' is a vector containing the ordering of variables at the final level of the tree. Essentially, this is the numeric ordering of the final labels. Due to the limit on the largest possible integer (overflow), the final labels can have at most 16 digits, i.e. the tree can have at most 16 levels. For large data sets, this may not be enough partitioning steps to result in final nodes (leaves) with only one variable each. Furthermore, PAM can not partition a node of size 3 or less, so that leaves may contain 2 or 3 variables regardless of the number of levels in the tree. Hence, the final ordering of variables is completed by ordering the variables in any leaf of size 2 or larger with the method determined by the value of `ord`: "own" (relative to own medoid), "neighbor" (relative to next medoid to the right), or "co" (maximize correlation ordering).

'medoids' is a matrix containing the labels and corresponding medoids for each internal node and leaf of the tree. The number of digits in the label indicates the level for that node. The medoid refers to a row of `data`

`call` the matched 'call' generating the HOPACH output

`metric` the distance metric

Note

Thank you to Karen Vranizan <vranizan@uclink.berkeley.edu> for her input

Author(s)

Katherine S. Pollard <kpollard@gladstone.ucsf.edu> and Mark J. van der Laan <laan@stat.berkeley.edu>, with Greg Wall

References

van der Laan, M.J. and Pollard, K.S. A new algorithm for hybrid hierarchical clustering with visualization and the bootstrap. Journal of Statistical Planning and Inference, 2003, 117, pp. 275-303.

http://www.stat.berkeley.edu/~laan/Research/Research_subpages/Papers/hopach.pdf

http://www.bepress.com/ucbbiostat/paper107/

http://www.stat.berkeley.edu/~laan/Research/Research_subpages/Papers/jsmpaper.pdf

Kaufman, L. and Rousseeuw, P.J. (1990). Finding Groups in Data: An Introduction to Cluster Analysis. Wiley, New York.

See Also

distancematrix, labelstomss, boothopach, pam, makeoutput

Examples

```
#25 variables from two groups with 3 observations per variable
mydata<-rbind(cbind(rnorm(10,0,0.5),rnorm(10,0,0.5),rnorm(10,0,0.5)),cbind(rnorm(15,5,0.5
dimnames(mydata)<-list(paste("Var",1:25,sep=""),paste("Exp",1:3,sep=""))
mydist<-distancematrix(mydata,d="cosangle") #compute the distance matrix.

#clusters and final tree
clustresult<-hopach(mydata,dmat=mydist)
clustresult$clustering$k #number of clusters.
dimnames(mydata)[[1]][clustresult$clustering$medoids] #medoids of clusters.
table(clustresult$clustering$labels) #equal to clustresult$clustering$sizes.

#faster, sometimes fewer clusters
greedyresult<-hopach(mydata,clusters="greedy",dmat=mydist)

#only get the final ordering (no partitioning into clusters)
orderonly<-hopach(mydata,clusters="none",dmat=mydist)

#cluster the columns (rather than rows)
colresult<-hopach(t(mydata),dmat=distancematrix(t(mydata),d="euclid"))
```

---

hopach2tree *function to write MapleTree files for viewing hopach hierarchical*

Description

The MapleTree software (http://mapletree.sourceforge.net/) is an open source, cross-platform, visualization tool to graphically browse results of cluster analyses. The hopach2tree function takes a data matrix, plus corresponding hopach clustering output for genes and/or arrays, and writes the (.cdt, .gtr, and .atr) files needed to view these hierarchical clustering results in MapleTree. The function makeTree is called internally by hopach2tree.

Usage

```
hopach2tree(data, file = "HOPACH", hopach.genes = NULL, hopach.arrays = NULL,
dist.genes = NULL, dist.arrays = NULL, d.genes = "cosangle",
d.arrays = "euclid", gene.wts = NULL, array.wts = NULL, gene.names = NULL)

makeTree(labels, ord, medoids, dist, side = "GENE")
```

Arguments

data	data matrix, data frame or exprSet of gene expression measurements. Each column corresponds to an array, and each row corresponds to a gene. All values must be numeric. Missing values are ignored.
file	name for the output files (the extensions .cdt, .gtr and .atr will be added).
hopach.genes	output of the hopach function applied to genes (rows of data. If only arrays are clustered, hopach.genes can be NULL. There must be at least K=2 levels in the hopach final tree (ie: hopach.genes$final$labels greater than 1 digit) for a gtr file to be generated.
hopach.arrays	optional output of the hopach function applied to arrays (columns of data. There must be at least K=2 levels in the hopach final tree (ie: hopach.arrays$final$labels greater than 1 digit) for an atr file to be generated.
dist.genes	matrix of pair wise distances between all genes. All values must be numeric, and missing values are not allowed. If NULL, this matrix is computed using the metric specified by d.genes. Only needed if genes are clustered (hopach.genes!=NULL).
dist.arrays	matrix of pair wise distances between all arrays. All values must be numeric, and missing values are not allowed. If NULL, this matrix is computed using the metric specified by d.arrays. Only needed if arrays are clustered (hopach.arrays!=NULL).
d.genes	character string specifying the metric to be used for calculating dissimilarities between genes. The currently available options are "cosangle" (cosine angle or uncentered correlation distance), "abscosangle" (absolute cosine angle or absolute uncentered correlation distance), "euclid" (Euclidean distance), "abseuclid" (absolute Euclidean distance), "cor" (correlation distance), and "abscor" (absolute correlation distance). Advanced users can write their own distance functions and add these to the functions distancematrix() and distancevector().
d.arrays	character string specifying the metric to be used for calculating dissimilarities between arrays.
gene.wts	an optional vector of numeric weights for the genes.
array.wts	an optional vector of numeric weights for the arrays.
gene.names	optional vector of names or annotations for the genes, which can be different from the row names of data.
labels	final cluster labels from a hopach object.
ord	final ordering from a hopach object.
medoids	final medoids matrix from a hopach object.
dist	gene or array distance matrix.
side	character string specifying if the tree is for genes ("GENE", default) or arrays ("ARRY").

Value

The function hopach2tree has no value. It writes up to three text files to the current working directory. A .cdt file is always produced. This file can be used to visualize the data matrix as a heat map in MapleTree or other viewers such as TreeView (http://rana.lbl.gov/EisenSoftware.htm), jtreeview (http://sourceforge.net/projects/jtreeview/), and GeneXPress (http://genexpress.stanford.edu/).

When hopach.genes!=NULL, a .gtr is produced, and gene clustering results can be viewed, including ordering the genes in the heat map according to the final level of the hopach tree and drawing the dendogram for hierarchical gene clustering. Similarly, when hopach.arrays!=NULL, an .atr file is produced and array clustering results can be viewed.

The function makeTree is called internally by hopach2tree to make the objects needed to write the MapleTree files for a gene and/or array HOAPCH clustering result.

Warning

Operating systems use different end of line characters. These characters can cause errors in MapleTree when files generated on one OS are visualized on another OS. Hence, hopach2tree should be run on the same OS as MapleTree whenever possible.

Note

Thank you to Lisa Simirenko <lsimirenko@lbl.gov> for providing HOPACH views in MapleTree, and to Karen Vranizan <vranizan@uclink.berkeley.edu> for her input.

The MapleTree software can be downloaded from: http://sourceforge.net/projects/mapletree/

Author(s)

Katherine S. Pollard <kpollard@gladstone.ucsf.edu>

References

van der Laan, M.J. and Pollard, K.S. A new algorithm for hybrid hierarchical clustering with visualization and the bootstrap. Journal of Statistical Planning and Inference, 2003, 117, pp. 275-303.

http://www.stat.berkeley.edu/~laan/Research/Research_subpages/Papers/hopach.pdf

See Also

hopach, boothopach, bootmedoids, boot2fuzzy

Examples

```
#25 variables from two groups with 3 observations per variable
mydata<-rbind(cbind(rnorm(10,0,0.5),rnorm(10,0,0.5),rnorm(10,0,0.5)),cbind(rnorm(15,5,0.5
dimnames(mydata)<-list(paste("Var",1:25,sep=""),paste("Exp",1:3,sep=""))
mydist<-distancematrix(mydata,d="cosangle") #compute the distance matrix.

#clusters and final tree
clustresult<-hopach(mydata,dmat=mydist)

#write MapleTree files
hopach2tree(mydata,hopach.genes=clustresult,dist.genes=mydist)
```

`labelstomss` *Functions to compute silhouettes and split silhouettes*

Description

Silhouettes measure how well an element belongs to its cluster, and the average silhouette measures the strength of cluster membership overall. The Median (or Mean) Split Silhouette (MSS) is a measure of cluster heterogeneity. Given a partitioning of elements into groups, the MSS algorithm considers each group separately and computes the split silhouette for that group, which evaluates evidence in favor of further splitting the group. If the median (or mean) split silhouette over all groups in the partition is low, the groups are homogeneous.

Usage

```
labelstomss(labels, dist, khigh = 9, within = "med", between = "med",
hierarchical = TRUE)

msscheck(dist, kmax = 9, khigh = 9, within = "med", between = "med",
    force = FALSE, echo = FALSE, graph = FALSE)

silcheck(data, kmax = 9, diss = FALSE, echo = FALSE, graph = FALSE)
```

Arguments

`labels`	vector of cluster labels for each element in the set.
`dist`	numeric distance matrix containing the pair wise distances between all elements. All values must be numeric and missing values are not allowed.
`data`	a data matrix. Each column corresponds to an observation, and each row corresponds to a variable. In the gene expression context, observations are arrays and variables are genes. All values must be numeric. Missing values are ignored. In `silcheck`, `data` may also be a distance matrix or dissimilarity object if the argument `diss=TRUE`.
`khigh`	integer between 1 and 9 specifying the maximum number of children for each cluster when computing MSS.
`kmax`	integer between 1 and 9 specifying the maximum number of clusters to consider. Can be different from khigh, though typically these are the same value.
`within`	character string indicating how to compute the split silhouette for each cluster. The available options are "med" (median over all elements in the cluster) or "mean" (mean over all elements in the cluster).
`between`	character string indicating how to compute the MSS over all clusters. The available options are "med" (median over all clusters) or "mean" (mean over all clusters). Recommended to use the same value as `within`.
`hierarchical`	logical indicating if 'labels' should be treated as encoding a hierarchical tree, e.g. from HOAPCH.
`force`	indicator of whether to require at least 2 clusters, if FALSE (default), one cluster is considered.
`echo`	indicator of whether to print the selected number of clusters and corresponding MSS.

graph	indicator of whether to generate a plot of MSS (or average silhouette in silcheck) versus number of clusters.
diss	indicator of whether data is a dissimilarity matrix (or dissimilarity object), as in the pam function of the cluster package. If TRUE then data will be considered as a dissimilarity matrix. If FALSE, then data will be considered as a data matrix (observations by variables).

Details

The Median (and mean) Split Silhouette (MSS) criteria is defined in paper107 listed in the references (below). This criteria is based on the criteria function 'silhouette', proposed by Kaufman and Rousseeuw (1990). While average silhouette is a good global measure of cluster strength, MSS was developed to be more "aggressive" for finding small, homogeneous clusters in large data sets. MSS is a measure of average cluster homogeneity. The Median version is more robust than the Mean.

Value

For labelstomss, the median (or mean or combination) split silhouette, depending on the values of within and between.

For msscheck, a vector with first component the chosen number of clusters (minimizing MSS) and second component the corresponding MSS.

For silcheck, a vector with first component the chosen number of clusters (maximizing average silhouette) and second component the corresponding average silhouette.

Author(s)

Katherine S. Pollard <kpollard@gladstone.ucsf.edu> and Mark J. van der Laan <laan@stat.berkeley.edu>

References

http://www.bepress.com/ucbbiostat/paper107/

http://www.stat.berkeley.edu/~laan/Research/Research_subpages/Papers/jsmpaper.pdf

Kaufman, L. and Rousseeuw, P.J. (1990). Finding Groups in Data: An Introduction to Cluster Analysis. Wiley, New York.

See Also

pam, hopach, distancematrix

Examples

```
mydata<-rbind(cbind(rnorm(10,0,0.5),rnorm(10,0,0.5),rnorm(10,0,0.5)),cbind(rnorm(15,5,0.5
mydist<-distancematrix(mydata,d="cosangle") #compute the distance matrix.

#pam
result1<-pam(mydata,k=2)
result2<-pam(mydata,k=5)
labelstomss(result1$clust,mydist,hierarchical=FALSE)
labelstomss(result2$clust,mydist,hierarchical=FALSE)
```

```
#hopach
result3<-hopach(mydata,dmat=mydist)
labelstomss(result3$clustering$labels,mydist)
labelstomss(result3$clustering$labels,mydist,within="mean",between="mean")
```

`makeoutput` *function to write a text file with hopach output*

Description

The function `makeoutput` takes a data matrix and corresponding `hopach` clustering output, plus possibly bootstrap resampling output, and makes a table summarizing the clustering results. The table is written to a tab delimited text file.

Usage

```
makeoutput(data, hopachobj, bootobj = NULL, file = "HOPACH.out",
gene.names = NULL)
```

Arguments

`data`	data matrix, data frame or exprSet of gene expression measurements. Typically, each column corresponds to an array, and each row corresponds to a gene. For clustering arrays, the arrays appear in the rows and the genes in the columns. All values must be numeric. Missing values are ignored.
`hopachobj`	output of the `hopach` function.
`bootobj`	optional output of `boothopach` or `bootmedoids` - a matrix of bootstrap estimated cluster membership probabilities, with a row for each row in `data` and a column for each cluster.
`file`	filename for the table produced.
`gene.names`	optional names or annotations for the genes (arrays), which can be different from the row names of `data`

Details

The output table contains information about the rows of `data`: Index, UID and Name; the main cluster results: Cluster Number, Cluster Label (from `hopach`), and Cluster Level Order (the ordering of the elements in the level of the `hopach` tree at which the main clusters were identified); and the final level of the tree: Final Label, and Final Order (the ordering of elements in the final level of the tree). Sorting this table on Index results in the rows having the same order as in data. Sorting on Cluster Level Order results in the rows being ordered by cluster, and then within cluster based on the value of the argument `ord` to `hopach` (default is distance to the medoid). Sorting on Final Level Order results in the rows being ordered as in the leaves of the `hopach` tree, where clusters are still ordered and elements near each other in the ordering will have small pairwise distances.

Value

The function `makeoutput` has no value. It writes a tab delimited text file to the current working directory.

Note

Thank you to Karen Vranizan <vranizan@uclink.berkeley.edu> for helping to write this function.

Author(s)

Katherine S. Pollard <kpollard@gladstone.ucsf.edu>

References

van der Laan, M.J. and Pollard, K.S. A new algorithm for hybrid hierarchical clustering with visualization and the bootstrap. Journal of Statistical Planning and Inference, 2003, 117, pp. 275-303.

http://www.stat.berkeley.edu/~laan/Research/Research_subpages/Papers/hopach.pdf

See Also

hopach, boothopach, bootmedoids

Examples

```
#25 variables from two groups with 3 observations per variable
mydata<-rbind(cbind(rnorm(10,0,0.5),rnorm(10,0,0.5),rnorm(10,0,0.5)),cbind(rnorm(15,5,0.5
dimnames(mydata)<-list(paste("Var",1:25,sep=""),paste("Exp",1:3,sep=""))
mydist<-distancematrix(mydata,d="cosangle") #compute the distance matrix.

#clusters and final tree
clustresult<-hopach(mydata,dmat=mydist)

#bootstrap resampling
myobj<-boothopach(mydata,clustresult)

#write output file
makeoutput(mydata,clustresult,myobj)
```

---

prune *function to prune a HOPACH tree.*

Description

The hopach clustering function identifies a level of the tree with minimum MSS as the main clusters and also runs the tree down all the way to the final level. The prune function allows one to access a level of the tree other than the main clusters or the final level.

Usage

```
prune(data,hobj,level,dmat=NULL,ord="own")
```

Arguments

data	data matrix, data frame or exprSet of gene expression measurements. Typically, each column corresponds to an array, and each row corresponds to a gene. For clustering arrays, the arrays appear in the rows and the genes in the columns. All values must be numeric. Missing values are ignored.
hobj	output of the hopach function.
level	an integer specifying the level to which the tree should be pruned - can be greater than or less than the level with the main clusters.
dmat	matrix of pair wise distances between all genes (arrays). All values must be numeric. If NULL, this matrix is computed using the metric specified by the 'metric' given in hobj. If a matrix is provided, the user is responsible for ensuring that the metric used agrees with that used in computing hobj.
ord	character string indicating which of the two orderings produced by hopach should be used for the plot. If ord="final", the ordering of elements in the final level of the hierarchical tree is used. If ord="cluster", the ordering from the level of the tree corresponding to the main clusters is used. In both cases, the elements from each cluster will be contiguous. If ord="final", then the medoid element will appear in the middle of each cluster. Else, the ordering depends on the value of ord passed to the hopach function. If ord="none", then the elements are plotted in the same order as in dist.

Value

A list with the same components as are returned by the hopach function. The clustering component now contains the specified level instead of the main clusters.

Author(s)

Katherine S. Pollard <kpollard@gladstone.ucsf.edu> and Mark J. van der Laan <laan@stat.berkeley.edu>

References

van der Laan, M.J. and Pollard, K.S. A new algorithm for hybrid hierarchical clustering with visualization and the bootstrap. Journal of Statistical Planning and Inference, 2003, 117, pp. 275-303.

http://www.stat.berkeley.edu/~laan/Research/Research_subpages/Papers/hopach.pdf

See Also

hopach, makeoutput

Examples

```
mydata<-matrix(rnorm(600),nrow=100)
mydist<-distancematrix(mydata,d="cosangle")
clustresult<-hopach(mydata,dmat=mydist)
level2<-prune(mydata,clustresult,level=2,dmat=mydist,ord="own")
clustresult$clustering$k
level2$clustering$k
```

Index