BootstrapClusterTest *The BootstrapClusterTest Class*

Description

Performs a nonparametric bootstrap (sampling with replacement) test to determine whether the clusters found by an unsupervised method appear to be robust in a given data set.

Usage

```
BootstrapClusterTest(data, FUN, subsetSize, nTimes = 100, verbose = TRUE, ...)
```

Arguments

`data`	A data matrix, numerical data frame, or `ExpressionSet` object.
`FUN`	A `function` that, given a data matrix, returns a vector of cluster assignments. Examples of functions with this behavior are `cutHclust`, `cutKmeans`, `cutPam`, and `cutRepeatedKmeans`.
`...`	Additional arguments passed to the classifying function, `FUN`.
`subsetSize`	An optional integer argument. If present, each iteration of the bootstrap selects `subsetSize` rows from the original data matrix. If missing, each bootstrap contains the same number of rows as the original data matrix.
`nTimes`	The number of bootstrap samples to collect.
`verbose`	A logical flag

Objects from the Class

Objects should be created using the `BootstrapClusterTest` function, which performs the requested bootstrap on the clusters. Following the standard R paradigm, the resulting object can be summarized and plotted to determine the results of the test.

Slots

`f:` A `function` that, given a data matrix, returns a vector of cluster assignments. Examples of functions with this behavior are `cutHclust`, `cutKmeans`, `cutPam`, and `cutRepeatedKmeans`.

`subsetSize:` The number of rows to be included in each bootstrap sample.

`nTimes:` An integer, the number of bootstrap samples that were collected.

`call:` An object of class `call`, which records how the object was produced.

`result:` Object of class `matrix` containing, for each pair of columns in the original data, the number of times they belonged to the same cluster of a bootstrap sample.

Extends

Class `ClusterTest`, directly. See that class for descriptions of the inherited methods `image` and `hist`.

Methods

summary `signature(object = BootstrapClusterTest)`: Write out a summary of the object.

Author(s)

Kevin R. Coombes <kcoombes@mdanderson.org>

References

Kerr MK, Churchill GJ. Boostrapping cluster analysis: Assessing the reliability of conclusions from microarray experiments. PNAS 2001; 98:8961-8965.

See Also

`ClusterTest`, `PerturbationClusterTest`

Examples

```
# simulate data from two different groups
d1 <- matrix(rnorm(100*30, rnorm(100, 0.5)), nrow=100, ncol=30, byrow=FALSE)
d2 <- matrix(rnorm(100*20, rnorm(100, 0.5)), nrow=100, ncol=20, byrow=FALSE)
dd <- cbind(d1, d2)
cols <- rep(c('red', 'green'), times=c(30,20))
# peform your basic hierarchical clustering...
hc <- hclust(distanceMatrix(dd, 'pearson'), method='complete')

# bootstrap the clusters arising from hclust
bc <- BootstrapClusterTest(dd, cutHclust, nTimes=200, k=3, metric='pearson')
summary(bc)

# look at the distribution of agreement scores
hist(bc, breaks=101)

# let heatmap compute a new dendrogram from the agreement
image(bc, col=blueyellow(64), RowSideColors=cols, ColSideColors=cols)

# plot the agreement matrix with the original dendrogram
image(bc, dendrogram=hc, col=blueyellow(64), RowSideColors=cols, ColSideColors=cols)

# bootstrap the results of PAM
pamc <- BootstrapClusterTest(dd, cutPam, nTimes=200, k=3)
image(pamc, dendrogram=hc, col=blueyellow(64), RowSideColors=cols, ColSideColors=cols)

# contrast the behavior when all the data comes from the same group
xx <- matrix(rnorm(100*50, rnorm(100, 0.5)), nrow=100, ncol=50, byrow=FALSE)
hct <- hclust(distanceMatrix(xx, 'pearson'), method='complete')
bct <- BootstrapClusterTest(xx, cutHclust, nTimes=200, k=4, metric='pearson')
summary(bct)
image(bct, dendrogram=hct, col=blueyellow(64), RowSideColors=cols, ColSideColors=cols)

# cleanup
```

```
rm(d1, d2, dd, cols, hc, bc, pamc, xx, hct, bct)
```

GenePCA *The GenePCA Class*

Description

Perform principal components analysis on the genes (rows) from a microarray or proteomics experiment.

Usage

```
GenePCA(geneData)
## S4 method for signature 'GenePCA, missing':
plot(x, splitter=0)
```

Arguments

`geneData`	A data matrix, with rows interpreted as genes and columns as samples
`x`	a `GenePCA` object
`splitter`	A logical vector classifying the genes.

Details

This is a preliminary attenpt at a class for principal components analysis of genes, parallel to the `SamplePCA` class for samples. The interface will (one hopes) improve markedly in the next version of the library.

Value

The `GenePCA` function construcs and returns a valid object of the `GenePCA` class.

Objects from the Class

Objects should be created using the `GenePCA` function.

Slots

`scores:` A `matrix` of size PxN, where P is the number of rows and N the number fo columns in the input, representing the projections of the input rows onto the first N principal components.

`variances:` A `numeric` vector of length N; the amount of the total variance explained by each principal component.

`components:` A `matrix` of size NxN containing each of the first P principal components as columns.

Methods

plot `signature(x = GenePCA, y = missing)`: Plot the genes in the space of the first two principal components.

Author(s)

Kevin R. Coombes <kcoombes@mdanderson.org>

See Also

SamplePCA, princomp

Examples

```
# simulate samples from thre different groups, with generic genes
d1 <- matrix(rnorm(100*10, rnorm(100, 0.5)), nrow=100, ncol=10, byrow=FALSE)
d2 <- matrix(rnorm(100*10, rnorm(100, 0.5)), nrow=100, ncol=10, byrow=FALSE)
d3 <- matrix(rnorm(100*10, rnorm(100, 0.5)), nrow=100, ncol=10, byrow=FALSE)
dd <- cbind(d1, d2, d3)

# perform PCA in gene space
gpc <- GenePCA(dd)

# plot the results
plot(gpc)

# cleanup
rm(d1, d2, d3, dd, gpc)
```

Mosaic *The Mosaic Class*

Description

Produce "Eisen" plots of microarray or proteomics data.

Usage

```
Mosaic(data, sampleMetric = "pearson", sampleLinkage = "average", geneMetric = "euclid", geneLi

## S4 method for signature 'Mosaic':
pltree(x, colors, labels, ...)

## S4 method for signature 'Mosaic, missing':
plot(x, main=x@name, center=FALSE, limits=NULL,
 sampleColors=NULL, sampleClasses=NULL, geneColors=NULL, geneClasses=NULL, ...)
```

Arguments

`data`	Either a data frame or matrix with numeric values or an `ExpressionSet` as defined in the BioConductor tools for analyzing microarray data.
`sampleMetric`	Any valid distance metric that can be passed to the `distanceMatrix` function
`sampleLinkage`	Any valid linkage method that can be passed to the `hclust` function
`geneMetric`	Any valid distance metric that can be passed to the `distanceMatrix` function
`geneLinkage`	Any valid linkage method that can be passed to the `hclust` function
`center`	A logical flag; should the data rows be centered?
`usecor`	A logical flag; should the data rows be scaled to have standard deviation one?
`name`	A character string; the name of this object.
`x`	A `Mosaic` object.
`colors`	An optional vector of character strings containing color names to be used when labeling the trees in the dendrogram. If provided, then the length should equal the number of columns in the original data matrix.
`labels`	An optional vector of character strings used to label the leaves in the dendrogram. If omitted, the column names are used.
`main`	A character string; the plot title
`limits`	An numeric vector. If provided, the data is truncated for display purposes, both above and below, at the minimum and maximum values of `limits`.
`sampleColors`	An optional character vector containing colors that will be used to label different sample types with a color bar across the top of the heat map.
`sampleClasses`	A logical vector or factor used to classify the samples into groups. Alternatively, an integer specifying the number `k` of groups into which to cut the sample dendrogram.
`geneColors`	An optional character vector containing colors that will be used to label different gene types with a color bar along the side of the heat map.
`geneClasses`	A logical vector or factor used to classify the genes into groups. Alternatively, an integer specifying the number `k` of groups into which to cut the gene dendrogram.
`...`	Additional parameters for `heatmap`.

Details

One of the earliest papers in the microaray literature used independent clustering of the genes (rows) and samples (columns) to produce dendrograms that were plotted along with a red-green heat map of the centered expression values. Since that time, literally thousand of additional papers have published variations on these red-green images. R includes a function, `heatmap` that builds such figures. However, that function is general purpose and has numerous optional parameters to tweak the display. The purpose of the `Mosaic` class is to provide a simplified object-oriented wrapper around `heatmap`, which as a side benefit allows us to keep track of the distance metrics and linkage rules that were used to produce the resulting figure.

Value

The `Mosaic` function constructs and returns a valid object of the `Mosaic` class.

Objects from the Class

Objects should be created with the `Mosaic` function.

Slots

`data`: The `matrix` contaiing the numerical data

`samples`: A dendrogram of class `hclust` produced by clustering the biological samples (columns of `data`).

`genes`: A dendrogram of class `hclust` produced by clustering the genes (columns of `data`).

`sampleMetric`: A `character` string; the distance metric used to cluster the samples.

`sampleLinkage`: A `character` string; the linkage rule used to cluster the samples.

`geneMetric`: A `character` string; the distance metric used to cluster the genes.

`geneLinkage`: A `character` string; the linkage rule used to cluster the genes.

`call`: An object of class `call` recording how the object was constructed.

`name`: A `character` string; the name of this object.

Methods

plot `signature(x = Mosaic, y = missing)`: Produce the "Eisen" plot, using `heatmap`.

pltree `signature(x = Mosaic)`: Plot the sample class dendrogram in the object.

summary `signature(object = Mosaic)`: Write out a summary of the object.

Author(s)

Kevin R. Coombes <kcoombes@mdanderson.org>

References

Eisen MB, Spellman PT, Brown PO, Botstein D. Cluster analysis and display of genome-wide expression patterns. Proc Natl Acad Sci U S A. 1998 Dec 8;95(25):14863-8.

See Also

`heatmap`, `hclust`, `cutree`

Examples

```
# simulate data from three different sample groups
d1 <- matrix(rnorm(100*10, rnorm(100, 0.5)), nrow=100, ncol=10, byrow=FALSE)
d2 <- matrix(rnorm(100*10, rnorm(100, 0.5)), nrow=100, ncol=10, byrow=FALSE)
d3 <- matrix(rnorm(100*10, rnorm(100, 0.5)), nrow=100, ncol=10, byrow=FALSE)
dd <- cbind(d1, d2, d3)
kind <- factor(rep(c('red', 'green', 'blue'), each=10))
```

```
# prepare the Mosaic object
m <- Mosaic(dd, sampleMetric='pearson', geneMetric='spearman', center=TRUE, usecor=TRUE)
summary(m)

# The default plot with red-green color map
plot(m, col=redgreen(64))

# change to a blue-yellow color map, and mark the four top splits in the sample
# direction with a color bar along the top
plot(m, col=blueyellow(128), sampleClasses=4,
     sampleColors=c('red', 'green', 'blue', 'black'))

# This time, mark the three classes that we know are there
plot(m, col=blueyellow(128), sampleClasses=kind,
     sampleColors=c('red', 'green', 'blue'))

plot(m, col=blueyellow(128), geneClasses=3, geneColors=c('red', 'green', 'black'))

# In addition, mark the top 5 splits in the gene dendrogram
plot(m, col=blueyellow(128),
     sampleClasses=kind, sampleColors=c('red', 'green', 'black'),
     geneClasses=5, geneColors=c('cyan', 'magenta', 'royalblue', 'darkgreen', 'orange'))

# plot the sample dendrogram by itself
cols <- as.character(kind)
pltree(m, labels=1:30, colors=cols)

# cleanup
rm(d1, d2, d3, dd, kind, cols, m)
```

PCanova *The PCanova Class*

Description

Implements the PCANOVA method for determining whether a putative group structure is truly reflected in multivariate data set.

Usage

```
PCanova(data, classes, labels, colors, usecor = TRUE)

## S4 method for signature 'PCanova, missing':
plot(x, tag='', mscale=1, cex=1, ...)
```

Arguments

data	Either a data frame or matrix with numeric values or an ExpressionSet as defined in the BioConductor tools for analyzing microarray data.

`labels`	A character vector used to label points in plots. The length of the `labels` vector should equal the number of columns (samples) in the `data` matrix. Since only the first character of each label is used in the plots, these should be unique.
`classes`	A subset of the `labels` used to indicate distinct classes. Again, the method truncates each class indicator to a single letter.
`colors`	A character vector containing color names; this should be the same length as the vector of `labels`.
`usecor`	A logical flag; should the rows of the data matrix be standardized before use?
`x`	A `PCanova` object.
`tag`	A character string to name the object, used as part of the plot title.
`mscale`	A real number. This is a hack; for some reason, the projection of the sample vectors into the principal component space computed from the matrix of group means seems to be off by a factor approximately equal to the square root of the average number of samples per group. Until we sort out the corect formula, this term can be adjusted until the group means appear to be in the correct place in the plots.
`cex`	Character expansion factor used only in the plot legend on the plot of PC correlations.
`...`	Additional graphical parameters passed on to `plot` when displaying the principal components plots.

Details

The PCANOVA method was developed as part of the submission that won the award for best presentation at the 2001 conference on the Critical ASsessment of Microarray Data Analaysis (CAMDA; http://www.camda.duke.edu). The idea is to perform the equivalent of an analysis of variance (ANOVA) in principal component (PC) space. Let X(i,j) denote the jth column vector belonging to the ith group of samples. We can model this as X(i,j) = mu + tau(i) + E(i,j), where mu is the overall mean vector, tau(i) is the "effects" vector for the ith group, and E(i,j) is the vector of residual errors. We can perform principal components analysis on the full matrix X containing all the columns X(i,j), on the matrix containing all the group mean vectors mu + tau(i), and on the residual matrix containing all the E(i,j) vectors. PCANOVA develops a measure ("PC correlation") for comparing these three sets of principal components. If the PC correlation is close to 1, then two principal component bases are close together; if the PC correlation is close to zero, then two principal components bases are dissimilar. Strong group structures are recognizable because the PC correlation between the total-matrix PC space and the group-means PC space is much larger than the PC correlation between the total-matrix PC space and the residual PC space. Weak or nonexistent group structures are recognizable because the relative sizes of the PC correlations is reversed.

Value

The `PCanova` function returns an object of the `PCanova` class.

Objects from the Class

Objects should be created by calling the `PCanova` function.

Slots

`orig.pca`: A `matrix` containing the `scores` component from PCA performed on the total matrix. All principal components analyses are performed using the `SamplePCA` class.

`class.pca`: A `matrix` containing the `scores` component fromm PCA performed on the matrix of group-mean vectors.

`resid.pca`: A `matrix` containing the `scores` component rom PCA performed on the matrix of residuals.

`mixed.pca`: A `matrix` containing the projections of all the original vectoprs into the principal component space computed from the matrix of group mean vectors.

`xc`: An object produced by performing hierarchical clustering on the total data matrix, using `hclust` with pearson distance and average linkage.

`hc`: An object produced by performing hierarchical clustering on the matrix of group means, using `hclust` with pearson distance and average linkage.

`rc`: An object produced by performing hierarchical clustering on the matrix of residuals, using `hclust` with pearson distance and average linkage.

`n`: An integer; the number of samples.

`class2orig`: The `numeric` vector of PC correlations relating the total-matrix PCA to the group-means PCA.

`class2resid`: The `numeric` vector of PC correlations relating the residual PCA to the group-means PCA.

`orig2resid`: The `numeric` vector of PC correlations relating the total-matrix PCA to the residual PCA.

`labels`: A `character` vector of plot labels to indicate the group membership of samples.

`classes`: A `character` vector of labels identifying the distinct groups.

`colors`: A character vector of color names used to indicate the group membership fo samples in plots.

`call`: An object of class `call` that records hwo the object was constructed.

Methods

plot `signature(x = PCanova, y = missing)`: Plot the results of the PCANOVA test on the data. This uses `par` to set up a 2x2 layout of plots. The first three plots show the sample vectors (color-coded and labeled) in the space spanned by the first two principal components for each of the there PCAs. The final plot shows the three sets of PC correlations. Colors in the first three plots are determined by the `colors` slot of the object, which was set when the object was created. Colors in the PC correlation plot are determined by the current values of `COLOR.OBSERVED`, `COLOR.EXPECTED`, and `COLOR.PERMTEST`

pltree `signature(x = PCanova)`: Produce dendrograms of the three hierarchical clusters of the samples, based on all the data, the group means, and the residuals. Since this method uses `par` to put all three dendrograms in the same window, it cannot be combined with other plots.

summary `signature(object = PCanova)`: Write out a summary of the object.

BUGS

1 The projection of the sample vectors into the principal component space of the group-means is off by a scale factor. The `mscale` parameter provides a work-around.

[2] The pltree method fails if you only supply two groups; this may be a failure in `hclust` if you only provide two objects to cluster.

Author(s)

Kevin R. Coombes <kcoombes@mdanderson.org>

References

Examples of the output of PCANOVA applied to the NCI60 data set can be found at http://bioinformatics.mdanderson.org/camda01.html. The full description has not been published (out of laziness on the part of the author of this code). The only description that has appeared in print is an extremely brief description that can be found in the proceedings of the CAMDA 2001 conference.

See Also

SamplePCA

Examples

```
# simulate data from three groups
d1 <- matrix(rnorm(100*10, rnorm(100, 0.5)), nrow=100, ncol=10, byrow=FALSE)
d2 <- matrix(rnorm(100*10, rnorm(100, 0.5)), nrow=100, ncol=10, byrow=FALSE)
d3 <- matrix(rnorm(100*10, rnorm(100, 0.5)), nrow=100, ncol=10, byrow=FALSE)
dd <- cbind(d1, d2, d3)
# colors that match the groups
cols <- rep(c('red', 'green', 'blue'), each=10)

# compute the PCanova object
pan <- PCanova(dd, c('red', 'green', 'blue'), cols, cols)
summary(pan)

# view the PC plots
plot(pan)

# view the dendrograms
pltree(pan, line=-0.5)

# compare the results when there is no underlying group structure
dd <- matrix(rnorm(100*50, rnorm(100, 0.5)), nrow=100, ncol=50, byrow=FALSE)
```

```
cols <- rep(c('red', 'green', 'blue', 'orange', 'cyan'), each=10)
pan <- PCanova(dd, unique(cols), cols, cols)
plot(pan, mscale=1/sqrt(10))

pltree(pan, line=-0.5)

# cleanup
rm(d1, d2, d3, dd, cols, pan)
```

PerturbationClusterTest

The PerturbationClusterTest Class

Description

Performs a parametric bootstrap test (by adding independent Gaussian noise) to determine whether the clusters found by an unsupervised method appear to be robust in a given data set.

Usage

```
PerturbationClusterTest(data, FUN, nTimes = 100, noise = 1, verbose = TRUE, ...)
```

Arguments

`data`	A data matrix, numerical data frame, or `ExpressionSet` object.
`FUN`	A `function` that, given a data matrix, returns a vector of cluster assignments. Examples of functions with this behavior are `cutHclust`, `cutKmeans`, `cutPam`, and `cutRepeatedKmeans`.
`...`	Additional arguments passed to the classifying function, `FUN`.
`noise`	An optional numeric argument; the standard deviation of the mean zero Gaussian noise added to each measurement during each bootstrap. Defaults to 1.
`nTimes`	The number of bootstrap samples to collect.
`verbose`	A logical flag

Objects from the Class

Objects should be created using the `PerturbationClusterTest` function, which performs the requested bootstrap on the clusters. Following the standard R paradigm, the resulting object can be summarized and plotted to determine the results of the test.

Slots

f: A `function` that, given a data matrix, returns a vector of cluster assignments. Examples of functions with this behavior are `cutHclust`, `cutKmeans`, `cutPam`, and `cutRepeatedKmeans`.

noise: The standard deviation of the Gaussian noise added during each bootstrap sample.

nTimes: An intetger, the number of bootstrap samples that were collected.

call: An object of class `call`, which records how the object was produced.

result: Object of class `matrix` containing, for each pair of columns in the original data, the number of times they belonged to the same cluster of a bootstrap sample.

Extends

Class `ClusterTest`, directly. See that class for descriptions of the inherited methods `image` and `hist`.

Methods

summary `signature(object = PerturbationClusterTest)`: Write out a summary of the object.

Author(s)

Kevin R. Coombes <kcoombes@mdanderson.org>

References

Kerr MK, Churchill GJ. Boostrapping cluster analysis: Assessing the reliability of conclusions from microarray experiments. PNAS 2001; 98:8961-8965.

See Also

`ClusterTest`, `BootstrapClusterTest`

Examples

```
# simulate data from two different groups
d1 <- matrix(rnorm(100*30, rnorm(100, 0.5)), nrow=100, ncol=30, byrow=FALSE)
d2 <- matrix(rnorm(100*20, rnorm(100, 0.5)), nrow=100, ncol=20, byrow=FALSE)
dd <- cbind(d1, d2)
cols <- rep(c('red', 'green'), times=c(30,20))
# peform your basic hierarchical clustering...
hc <- hclust(distanceMatrix(dd, 'pearson'), method='complete')

# bootstrap the clusters arising from hclust
bc <- PerturbationClusterTest(dd, cutHclust, nTimes=200, k=3, metric='pearson')
summary(bc)

# look at the distribution of agreement scores
hist(bc, breaks=101)

# let heatmap compute a new dendrogram from the agreement
```

```
image(bc, col=blueyellow(64), RowSideColors=cols, ColSideColors=cols)

# plot the agreement matrix with the original dendrogram
image(bc, dendrogram=hc, col=blueyellow(64), RowSideColors=cols, ColSideColors=cols)

# bootstrap the results of K-means
kmc <- PerturbationClusterTest(dd, cutKmeans, nTimes=200, k=3)
image(kmc, dendrogram=hc, col=blueyellow(64), RowSideColors=cols, ColSideColors=cols)

# contrast the behavior when all the data comes from the same group
xx <- matrix(rnorm(100*50, rnorm(100, 0.5)), nrow=100, ncol=50, byrow=FALSE)
hct <- hclust(distanceMatrix(xx, 'pearson'), method='complete')
bct <- PerturbationClusterTest(xx, cutHclust, nTimes=200, k=4, metric='pearson')
summary(bct)
image(bct, dendrogram=hct, col=blueyellow(64), RowSideColors=cols, ColSideColors=cols)

# cleanup
rm(d1, d2, dd, cols, hc, bc, kmc, xx, hct, bct)
```

`SamplePCA` *The SamplePCA Class*

Description

Perform principal components analysis on the samples (columns) from a microarray or proteomics experiment.

Usage

```
SamplePCA(data, splitter = 0, usecor = FALSE, center = TRUE)
## S4 method for signature 'SamplePCA, missing':
plot(x, splitter=x@splitter, col, main='', which=1:2, ...)
```

Arguments

`data`	Either a data frame or matrix with numeric values or an `ExpressionSet` as defined in the BioConductor tools for analyzing microarray data.
`splitter`	If `data` is a data frame or matrix, then splitter must be either a logical vector or a factor. If `data` is an `ExpressionSet`, then `splitter` can be a character string that names one of the factor columns in the associated `phenoData` subobject.
`center`	A logical value; should the rows of the data matrix be centered first?
`usecor`	A logical value; should the rows of the data matrix be scaled to have standard deviation 1?
`x`	A `SamplePCA` object
`col`	A list of colors to represent each level of the `splitter` in the plot. If this parameter is missing, the function will select colors automatically.

`main`	A character string; the plot title
`which`	A numeric vector of length two specifying which two principal components should be included in the plot.
`...`	Additional graphical parameters for `plot`

.

Details

The main reason for developing the `SamplePCA` class is that the `princomp` function is very inefficient when the number of variables (in the microarray setting, genes) far exceeds the number of observations (in the microarray setting, biological samples). The `princomp` function begins by computing the full covariance matrix, which gets rather large in a study involving tens of thousands of genes. The `SamplePCA` class, by contrast, uses singular value decomposition (`svd`) on the original data matrix to compute the principal components.

Value

The `SamplePCA` function constructs and returns an object of the `SamplePCA` class. We assume that the input data matrix has N columns (of biological samples) and P rows (of genes).

The predict method returns a matrix whose size is the number of columns in the input by the number of principal components.

Objects from the Class

Objects should be created using the `SamplePCA` function. In the simplest case, you simply pass in a data matrix and a logical vector, `splitter`, assigning classes to the columns, and the constructor performs principal components analysis on the column. The `splitter` is ignored by the constructor and is simply saved to be used by the plotting routines. If you omit the `splitter`, then no grouping structure is used in the plots.

If you pass `splitter` as a factor instead of a logical vector, then the plotting routine will distinguish all levels of the factor. The code is likely to fail, however, if one of the levels of the factor has zero representatives among the data columns.

As with the class comparison functions (see, for example, `MultiTtest`) that are part of OOMPA, we can also perform PCA on `ExpressionSet` objects from the BioConductor libraries. In this case, we pass in an `ExpressionSet` object along with a character string containing the name of a factor to use for splitting the data.

Slots

`scores:` A `matrix` of size NxN, where N is the number of columns in the input, representing the projections of the input columns onto the first N principal components.

`variances:` A `numeric` vector of length N; the amount of the total variance explained by each principal component.

`components:` A `matrix` of size PxN (the same size as the input matrix) containing each of the first P principal components as columns.

`splitter:` A logical vector or factor of length N classifying the coluimns into known groups.

`usecor:` A `logical` value; was the data standardized?

`shift:` A `numeric` vector of length P; the mean vector of the input data, which is used for centering by the `predict` method.

`scale:` A `numeric` vector of length P; the standard deviation of the input data, which is used for scaling by the `predict` method.

`call:` An object of class `call` that records how the object was created.

Methods

plot `signature(x = SamplePCA, y = missing)`: Plot the samples in a two-dimensional principal component space.

predict `signature(object = SamplePCA)`: Project new data into the principal component space.

screeplot `signature(x = SamplePCA)`: Produce a bar chart of the variances explained by each principal component.

summary `signature(object = SamplePCA)`: Write out a summary of the object.

Author(s)

Kevin R. Coombes <kcoombes@mdanderson.org>

See Also

princomp, GenePCA

Examples

```
# simulate datda from three different groups
d1 <- matrix(rnorm(100*10, rnorm(100, 0.5)), nrow=100, ncol=10, byrow=FALSE)
d2 <- matrix(rnorm(100*10, rnorm(100, 0.5)), nrow=100, ncol=10, byrow=FALSE)
d3 <- matrix(rnorm(100*10, rnorm(100, 0.5)), nrow=100, ncol=10, byrow=FALSE)
dd <- cbind(d1, d2, d3)
kind <- factor(rep(c('red', 'green', 'blue'), each=10))

# perform PCA
spc <- SamplePCA(dd, splitter=kind)

# plot the results
plot(spc, col=levels(kind))

# mark the group centers
x1 <- predict(spc, matrix(apply(d1, 1, mean), ncol=1))
points(x1[1], x1[2], col='red', cex=2)
x2 <- predict(spc, matrix(apply(d2, 1, mean), ncol=1))
points(x2[1], x2[2], col='green', cex=2)
x3 <- predict(spc, matrix(apply(d3, 1, mean), ncol=1))
points(x3[1], x3[2], col='blue', cex=2)

# check out the variances
screeplot(spc)
```

```
# cleanup
rm(d1, d2, d3, dd,kind, spc, x1, x2, x3)
```

`aspectHeatmap` *Heatmap with control over the aspect ratio*

Description

This function replaces the heatmap function in the stats package with a function with additional features. First, the user can specify an aspect ratio instead of being forced to accept a square image even when the matrix is not square. Second, the user can overlay horizontal or vertical lines to mark out important regions.

Usage

```
aspectHeatmap(x, Rowv = NULL, Colv = if (symm) "Rowv" else NULL, distfun = dist, hclustfun = hc
```

Arguments

`x`	See documentation for heatmap.
`Rowv`	See documentation for heatmap.
`Colv`	See documentation for heatmap.
`distfun`	See documentation for heatmap.
`hclustfun`	See documentation for heatmap.
`reorderfun`	See documentation for heatmap.
`add.expr`	See documentation for heatmap.
`symm`	See documentation for heatmap.
`revC`	See documentation for heatmap.
`scale`	See documentation for heatmap.
`na.rm`	See documentation for heatmap.
`margins`	See documentation for heatmap.
`ColSideColors`	See documentation for heatmap.
`RowSideColors`	See documentation for heatmap.
`cexRow`	See documentation for heatmap.
`cexCol`	See documentation for heatmap.
`labRow`	See documentation for heatmap.
`labCol`	See documentation for heatmap.
`main`	See documentation for heatmap.
`xlab`	See documentation for heatmap.
`ylab`	See documentation for heatmap.

`keep.dendro`	See documentation for heatmap.
`verbose`	See documentation for heatmap.
`hExp`	Height expansion factor; default = 1.
`wExp`	Width expansion factor; default = 1.
`vlines`	Vector of positions at which to draw vertical lines.
`hlines`	Vector of positions at which to draw horizontal lines.
`lasCol`	Axis label style (las) for columns.
`...`	Additional arguments passed along to the image command.

Details

The new arguments `hExp` and `wExp` are "expansion factors" for the height and width of the figure, respectively. They are used to alter the arguments passed to the layout function internally to heatmap.

The new arguments `hlines` and `vlines` are used to specify points at which horizontal or vertical lines, respectively, should be overlaid on the image.

Value

The same as the heatmap function.

Author(s)

Kevin R. Coombes <kcoombes@mdanderson.org>

See Also

heatmap

Examples

```
nC <- 30
nR <- 1000
fakeData <- matrix(rnorm(nC*nR), ncol=nC, nrow=nR)
aspectHeatmap(fakeData, scale='none', hExp=2, wExp=1.4, margins=c(6,3))
aspectHeatmap(fakeData, scale='none', hExp=2, wExp=1.4, margins=c(6,3), vlines=15.5, hlines=c(100, 300))
```

`cluster3` *Cluster a Dataset Three Ways*

Description

Produces and plots dendrograms using three similarity measures: Euclidean distance, Pearson correlation, and Manhattan distance on dichotomized data.

Usage

```
cluster3(data, eps = logb(1, 2), name = "", labels = dimnames(data)[[2]])
```

Arguments

`data`	A matrix, numeric data.frame, or `ExpressionSet` object.
`eps`	A numerical value; the threshold at which to dichotomize the data
`name`	A character string to label the plots
`labels`	A vector of character strings used to label the items in the dendrograms.

Value

Invisibly returns the `data` object on which it was invoked.

Author(s)

Kevin R. Coombes <kcoombes@mdanderson.org>

See Also

`hclust`, `plclust`

Examples

```
# simulate data from two different classes
d1 <- matrix(rnorm(100*30, rnorm(100, 0.5)), nrow=100, ncol=30, byrow=FALSE)
d2 <- matrix(rnorm(100*20, rnorm(100, 0.5)), nrow=100, ncol=20, byrow=FALSE)
dd <- cbind(d1, d2)
# cluster it 3 ways
par(mfrow=c(2,2))
cluster3(dd)
par(mfrow=c(1,1))
```

`ClusterTest-class` *The ClusterTest Class*

Description

This is a base class for tests that attempt to determine whether the groups found by an unsupervised clustering method are statistically significant.

Usage

```
## S4 method for signature 'ClusterTest':
image(x, dendrogram, ...)
```

Arguments

`x`	An object of the `ClusterTest` class.
`dendrogram`	An object with S3 class `hclust`, as returned by the `hclust` function.
`...`	Additional graphical parameters to be passed to the standard `image` function.

Objects from the Class

Objects can be created by calls of the form `new("ClusterTest", ...)`. Most users, however, will only create objects from one of the derived classes such as `BootstrapClusterTest` or `PerturbationClusterTest`.

Slots

`call`: An object of class `call`, which shows how the object was constructed.

`result`: A symmetric `matrix` containing the results of the cluster reproducibility test. The size of the matrix corresponds to the number of samples (columns) in the data set on which the test was performed. The `result` matrix should contain "agreement" values between 0 and 1, representing for each pair of samples the fraction of times that they were collected into the same cluster.

Methods

hist `signature(x = "ClusterTest")`: Produces a histogram of the agreement fractions. When a true group structure exists, one expects a multimodal distribution,with one group of agreements near 0 (for pairs belonging to different clusters) and one group of agreements near 1 (for pairs belonging to the same cluster).

image `signature(x = "ClusterTest")`: Uses the `heatmap` function to display the agreement matrix. The optional `dendrogram` argument should be used to display the extent to which the ageement matrix matches the results of hierarchical clustering using the full data set.

summary `signature(object = "ClusterTest")`: Write out a summary of the object.

Author(s)

Kevin R. Coombes <kcoombes@mdanderson.org>

References

Kerr MK, Churchill GJ. Boostrapping cluster analysis: Assessing the reliability of conclusions from microarray experiments. PNAS 2001; 98:8961-8965.

See Also

`BootstrapClusterTest`, `PerturbationClusterTest`, `heatmap`

Examples

```
# simulate data from two different classes
d1 <- matrix(rnorm(100*30, rnorm(100, 0.5)), nrow=100, ncol=30, byrow=FALSE)
d2 <- matrix(rnorm(100*20, rnorm(100, 0.5)), nrow=100, ncol=20, byrow=FALSE)
dd <- cbind(d1, d2)

# cluster the data
hc <- hclust(distanceMatrix(dd, 'pearson'), method='average')

# make a fake reproducibility matrix
fraud <- function(x) {
  new('ClusterTest', result=abs(cor(x)), call=match.call())
}

fake <- fraud(dd)
summary(fake)

hist(fake)

image(fake) # let heatmap compute a new dendrogram from the agreements

image(fake, dendrogram=hc) # use the actual dendrogram from the data

image(fake, dendrogram=hc, col=blueyellow(64)) # change the colors

#cleanup
rm(fake, fraud, hc, dd, d1, d2)
```

---

colorschemes *Color Schemes for Images and Heat Maps*

Description

Create a vector of `N` contiguous colors.

Usage

```
redscale(N)
greenscale(N)
bluescale(N)
blueyellow(N)
redgreen(N)
jetColors(N)
```

Arguments

`N`	an integer; the number of distinct levels in the color map

Details

The color maps that ship with R (see, for example, `terrain.colors`) do not include the most common colors maps used in publications in the microarray literature. This collection of color maps expands the available options. The functions `redscale`, `greenscale`, and `bluescale` each range from pure black for low values to a pure primary color for high values.

The `redgreen` color map ranges from pure green at the low end, through black in the middle, to pure red at the high end. Although this is the most common color map used in the microarray literature, it will prove problematic for individuals with red-green color-blindness.

The `blueyellow` color map ranges from pure blue at the low end, through gray in the middle, to pure yellow at the high end.

The `jetColors` map tries to reproduce the default "jet" color map from MATLAB.

Value

A character vector '`cv`' of color names. This can be used to create a user-defined color palette for subsequent graphics by '`palette(cv)`' or directly in a '`col=`' specification in '`par`' or in graphics functions such as '`image`' or '`heatmap`'.

BUGS

The names `redgreen` and `blueyellow` are inconsistent with respect to which color represents low values and which color represents high values. It is too late to fix this.

Author(s)

Kevin R. Coombes <kcoombes@mdanderson.org>

See Also

`rainbow`, `topo.colors`, `terrain.colors`, `heat.colors`, `rgb`, `image`, `heatmap`, `palette`.

Examples

```
data <- matrix(1:1024, nrow=1024)
image(data, col=bluescale(64))
image(data, col=redgreen(32))
image(data, col=redscale(128))
image(data, col=blueyellow(64))
image(data, col=jetColors(64))
rm(data) # cleanup
```

Description

This function computes and returns the distance matrix determined by using the specified distance metric to compute the distances between the columns fo a data matrix.

Usage

```
distanceMatrix(dataset, metric, ...)
```

Arguments

`dataset`	A numeric matrix or an `ExpressionSet`
`metric`	A character string defining the distance metric. This can be `pearson`, `sqrt pearson`, `spearman`, `absolute pearson`, `weird` or any of the metrics accepted by the `dist` function. At present, the latter function accepts `euclidean`, `maximum`, `manhattan`, `canberra`, `binary`, or `minkowski`. Any initial substring that uniquely defines one of the metrics will work.
`...`	Additional parameters to be passed on to `dist`.

Details

This function differs from `dist` in two ways, both of which are motivated by common practice in the analysis of microarray or proteomics data. First, it computes distances between column vectors instead of between row vectors. In a typical microarray experiment, the data is organized so the rows represent genes and the columns represent different biological samples. In many applications, relations between the biological samples are more interesting than relationships between genes. Second, `distanceMatrix` adds additional distance metrics based on correlation.

`pearson` The most common metric used in the microarray literature is the `pearson` distance, which can be computed in terms of the Pearson correlation coefficient as `(1-cor(dataset))/2`.

`spearman` The `spearman` metric used the same formula, but substitutes the Spearman rank correlation for the Pearson correlation.

`absolute pearson` The `absolute pearson` metric used the absolute correlation coefficient; i.e., `(1-abs(cor(dataset)))`.

`sqrt pearson` The `sqrt pearson` metric used the square root of the pearson distance metric; i.e., `sqrt(1-cor(dataset))`.

`weird` The `weird` metric uses the Euclidean distance between the vectors of correlation coefficients; i.e., dist(cor(dataset)).

Value

A distance matrix in the form of an object of class `dist`, of the sort returned by the `dist` function or the `as.dist` function.

BUGS

It would be good to have a better name for the `weird` metric.

Author(s)

Kevin R. Coombes <kcoombes@mdanderson.org>

See Also

`dist`, `as.dist`

Examples

```
dd <- matrix(rnorm(100*5, rnorm(100)), nrow=100, ncol=5)
distanceMatrix(dd, 'pearson')
distanceMatrix(dd, 'euclid')
distanceMatrix(dd, 'sqrt')
distanceMatrix(dd, 'weird')
rm(dd) # cleanup
```

`hclust` *An S4 hclust class*

Description

Creates an S4 hclust class that is equivalent to the S3 hclust class returned by the hclust function.

Author(s)

Kevin R. Coombes <kcoombes@mdanderson.org>

See Also

`heatmap`, `hclust`, `cutree`

Description

Unsupervised clustering algorithms, such as partitioning around medoids (pam), K-means (kmeans), or hierarchical clustering (hclust) after cutting the tree, produce a list of class assignments along with other structure. To simplify the interface for the BootstrapClusterTest and PerturbationClusterTest, we have written these routines that simply extract these cluster assignments.

Usage

```
cutHclust(data, k, method = "average", metric = "pearson")
cutPam(data, k)
cutKmeans(data, k)
cutRepeatedKmeans(data, k, nTimes)

repeatedKmeans(data, k, nTimes)
```

Arguments

data	A numerical data matrix
k	The number of classes desired from the algorithm
method	Any valid linkage method that can be passed to the hclust function
metric	Any valid distance metric that can be passed to the distanceMatrix function
nTimes	An integer; the number of times to repeat the K-means algorithm with a different random starting point

Details

Each of the clustering routines used here has a different structure for storing cluster assignments. The kmeans function stores the assignments in a 'cluster' attribute. The pam function uses a 'clustering' attribute. For hclust, the assigments are produced by a call to the cutree function.

It has been observed that the K-means algorithm can converge to different solutions depending on the starting values of the group centers. We also include a routine (repeatedKmeans) that runs the K-means algorithm repeatedly, using different randomly generated staring points each time, saving the best results.

Value

Each of the cut... functions returns a vector of integer values representing the cluster assignments found by the algorithm.

The `repeatedKmeans` function returns a list `x` with three components. The component `x$kmeans` is the result of the call to the `kmeans` function that produced the best fit to the data. The component `x$centers` is a matrix containing the list of group centers that were used in the best call to `kmeans`. The component `x$withinss` contains the sum of the within-group sums of squares, which is used as the measure of fitness.

Author(s)

Kevin R. Coombes <kcoombes@mdanderson.org>

See Also

pam, hclust, cutree, kmeans.

Examples

```
# simulate data from three different groups
d1 <- matrix(rnorm(100*10, rnorm(100, 0.5)), nrow=100, ncol=10, byrow=FALSE)
d2 <- matrix(rnorm(100*10, rnorm(100, 0.5)), nrow=100, ncol=10, byrow=FALSE)
d3 <- matrix(rnorm(100*10, rnorm(100, 0.5)), nrow=100, ncol=10, byrow=FALSE)
dd <- cbind(d1, d2, d3)

cutKmeans(dd, k=3)
cutKmeans(dd, k=4)

cutHclust(dd, k=3)
cutHclust(dd, k=4)

cutPam(dd, k=3)
cutPam(dd, k=4)

cutRepeatedKmeans(dd, k=3, nTimes=10)
cutRepeatedKmeans(dd, k=4, nTimes=10)

# cleanup
rm(d1, d2, d3, dd)
```

---

plotColoredClusters *Plot Dendrograms with Color-Coded Labels*

Description

Provides an interface to plclust that makes it easier to plot dendrograms with labels that are color-coded, usually to indicate the different levels of a factor.

Usage

```
plotColoredClusters(hd, labs, cols, cex = 0.8, main = "", line = 0, ...)
```

Arguments

`hd`	An object with S3 class `hclust`, as produced by the `hclust` function.
`labs`	A vector of character strings used to label the leaves in the dendrogram
`cols`	A vector of color names suitable for passing to the `col` argument of graphics routines.
`cex`	A numeric value; the character expansion parameter of `par`.
`main`	A character string; the plot title
`line`	An integer determining how far away to plot the labels; see `mtext` for details.
`...`	Any additional graphical parameters that can be supplied when plotting an `hclust` object.

Details

This function is used to implement the `pltree` methods of the `Mosaic` class and the `PCanova` class. It simply bundles a two step process (first plotting the dendrogram with no labels, followed by writing the labels in the right places with the desired colors) into a single unit.

Value

The function has no useful return value; it merely produces a plot.

See Also

`plclust`, `hclust`, `Mosaic`, `PCanova`, `par`

Examples

```
# simulate data from three different groups
d1 <- matrix(rnorm(100*10, rnorm(100, 0.5)), nrow=100, ncol=10, byrow=FALSE)
d2 <- matrix(rnorm(100*10, rnorm(100, 0.5)), nrow=100, ncol=10, byrow=FALSE)
d3 <- matrix(rnorm(100*10, rnorm(100, 0.5)), nrow=100, ncol=10, byrow=FALSE)
dd <- cbind(d1, d2, d3)

# perform hierarchical clustering using correlation
hc <- hclust(distanceMatrix(dd, 'pearson'), method='average')
cols <- rep(c('red', 'green', 'blue'), each=10)
labs <- paste('X', 1:30, sep='')

# plot the dendrogram with color-coded groups
plotColoredClusters(hc, labs=labs, cols=cols)

#cleanup
rm(d1, d2, d3, dd, hc, cols, labs)
```