Package: moduleColor (via r-universe)

October 25, 2024

Version 1.8-4

Date 2022-04-08

Title Basic Module Functions

Author Peter Langfelder <Peter.Langfelder@gmail.com> and Steve Horvath <SHorvath@mednet.ucla.edu>

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

Depends R (>= 2.3.0), stats, impute, grDevices, dynamicTreeCut

ZipData no

License GPL (>= 2)

Description Methods for color labeling, calculation of eigengenes, merging of closely related modules.

URL https://horvath.genetics.ucla.edu/html/CoexpressionNetwork/BranchCutting/

NeedsCompilation no

Date/Publication 2022-04-09 16:02:29 UTC

Repository https://plangfelder.r-universe.dev

RemoteUrl https://github.com/cran/moduleColor

RemoteRef HEAD

RemoteSha fc6c38598a7d76e10a33c46255e94577213f5d3e

Contents

`moduleColor-package` *Basic Module Functions*

Description

Methods for color labeling, calculation of eigengenes, merging of closely related modules.

Details

Package:	moduleColor
Version:	1.08-3
Date:	2014-11-25
Depends:	R, stats, impute, grDevices, dynamicTreeCut
ZipData:	no
License:	GPL version 2 or newer
URL:	http://www.genetics.ucla.edu/labs/horvath/CoexpressionNetwork/BranchCutting/

Index:

```
checkSets               Retrieve basic sizes of a group of datasets.
collectGarbage          Iterative garbage collection.
consensusMEDissimilarity
                        Consensus dissimilarity of module eigengenes.
consensusOrderMEs       Put close eigenvectors next to each other in
                        several sets.
fixDataStructure        Put single-set data into a form useful for
                        multiset calculations.
labels2colors           Convert numerical labels to colors.
mergeCloseModules       Merge close modules of gene expression data.
moduleColor-package     Basic module functions.
moduleColor.getMEprefix Get the prefix used to label module eigengenes.
moduleColor.version Returns the version number of the package.
```

```
moduleColor.revisionDate
   Returns the revision date of the package.
moduleEigengenes       Calculate module eigengenes.
moduleNumber           Fixed-height cut of a dendrogram.
multiSetMEs            Calculate module eigengenes.
normalizeLabels        Transform numerical labels into normal order.
orderMEs               Put close eigenvectors next to each other
plotHclustColors       Plot color bars corresponding to modules
removeGreyME  Remove the grey module eigengene from given eigengenes.
standardColors         Colors this library uses for labeling modules.
```

Author(s)

Peter Langfelder <Peter.Langfelder@gmail.com> and Steve Horvath <SHorvath@mednet.ucla.edu>

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

`checkSets` *Check structure and retrieve sizes of a group of datasets.*

Description

Checks whether given sets have the correct format and retrieves dimensions.

Usage

```
checkSets(data, checkStructure = FALSE, useSets = NULL)
```

Arguments

`data`	A vector of lists; in each list there must be a component named `data` whose content is a matrix or dataframe or array of dimension 2.
`checkStructure`	If `FALSE`, incorrect structure of `data` will trigger an error. If `TRUE`, an appropriate flag (see output) will be set to indicate whether `data` has correct structure.
`useSets`	Optional specification of entries of the vector `data` that are to be checked. Defaults to all components. This may be useful when `data` only contains information for some of the sets.

Details

For multiset calculations, many quantities (such as expression data, traits, module eigengenes etc) are presented by a common structure, a vector of lists (one list for each set) where each list has a component `data` that contains the actual (expression, trait, eigengene) data for the corresponding set in the form of a dataframe. This funtion checks whether `data` conforms to this convention and retrieves some basic dimension information (see output).

Value

A list with components

nSets	Number of sets (length of the vector data).
nGenes	Number of columns in the data components in the lists. This number must be the same for all sets.
nSamples	A vector of length nSets giving the number of rows in the data components.
structureOK	Only set if the argument checkStructure equals TRUE. The value is TRUE if the paramter data passes a few tests of its structure, and FALSE otherwise. The tests are not exhaustive and are meant to catch obvious user errors rather than be bulletproof.

Author(s)

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

---

collectGarbage *Iterative garbage collection.*

Description

Performs garbage collection until free memory idicators show no change.

Usage

```
collectGarbage()
```

Value

None.

Author(s)

Steve Horvath

consensusMEDissimilarity

Consensus dissimilarity of module eigengenes.

Description

Calculates consensus dissimilarity (1-cor) of given module eigengenes relaized in several sets.

Usage

```
consensusMEDissimilarity(MEs, useAbs = FALSE, useSets = NULL, method = "consensus")
```

Arguments

MEs	Module eigengenes of the same modules in several sets.
useAbs	Controls whether absolute value of correlation should be used instead of correlation in the calculation of dissimilarity.
useSets	If the consensus is to include only a selection of the given sets, this vector (or scalar in the case of a single set) can be used to specify the selection. If NULL, all sets will be used.
method	A character string giving the method to use. Allowed values are (abbreviations of) "consensus" and "majority". The consensus dissimilarity is calculated as the minimum of given set dissimilarities for "consensus" and as the average for "majority".

Details

This function calculates the individual set dissimilarities of the given eigengenes in each set, then takes the (parallel) maximum or average over all sets. For details on the structure of imput data, see checkSets.

Value

A dataframe containing the matrix of dissimilarities, with names and rownames set appropriately.

Author(s)

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

See Also

checkSets

`consensusOrderMEs` *Put close eigenvectors next to each other in several sets.*

Description

Reorder given (eigen-)vectors such that similar ones (as measured by correlation) are next to each other. This is a multi-set version of `orderMEs`; the dissimilarity used can be of consensus type (for each pair of eigenvectors the consensus dissimilarity is the maximum of individual set dissimilarities over all sets) or of majority type (for each pair of eigenvectors the consensus dissimilarity is the average of individual set dissimilarities over all sets).

Usage

```
consensusOrderMEs(MEs, useAbs = FALSE, useSets = NULL,
                  greyLast = TRUE,
                  greyName = paste(moduleColor.getMEprefix(), "grey", sep=""),
                  method = "consensus")
```

Arguments

`MEs`	Module eigengenes of several sets in a multi-set format (see `checkSets`). A vector of lists, with each list corresponding to one dataset and the module eigengenes in the component `data`, that is `MEs[[set]]$data[sample, module]` is the expression of the eigengene of module `module` in sample `sample` in dataset `set`. The number of samples can be different between the sets, but the modules must be the same.
`useAbs`	Controls whether vector similarity should be given by absolute value of correlation or plain correlation.
`useSets`	Allows the user to specify for which sets the eigengene ordering is to be performed.
`greyLast`	Normally the color grey is reserved for unassigned genes; hence the grey module is not a proper module and it is conventional to put it last. If this is not desired, set the parameter to `FALSE`.
`greyName`	Name of the grey module eigengene.
`method`	A character string giving the method to be used calculating the consensus dissimilarity. Allowed values are (abbreviations of) `"consensus"` and `"majority"`. The consensus dissimilarity is calculated as the maximum of given set dissimilarities for `"consensus"` and as the average for `"majority"`.

Details

Ordering module eigengenes is useful for plotting purposes. This function calculates the consensus or majority dissimilarity of given eigengenes over the sets specified by `useSets` (defaults to all sets). A hierarchical dendrogram is calculated using the dissimilarity and the order given by the dendrogram is used for the eigengenes in all other sets.

Value

A vector of lists of the same type as MEs containing the re-ordered eigengenes.

Author(s)

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

See Also

moduleEigengenes, multiSetMEs, orderMEs

---

fixDataStructure *Put single-set data into a form useful for multiset calculations.*

Description

Encapsulates single-set data in a wrapper that makes the data suitable for functions working on multiset data collections.

Usage

```
fixDataStructure(data, verbose = 0, indent = 0)
```

Arguments

data	A dataframe, matrix or array with two dimensions to be encapsulated.
verbose	Controls verbosity. 0 is silent.
indent	Controls indentation of printed progress messages. 0 means no indentation, every unit adds two spaces.

Details

For multiset calculations, many quantities (such as expression data, traits, module eigengenes etc) are presented by a common structure, a vector of lists (one list for each set) where each list has a component data that contains the actual (expression, trait, eigengene) data for the corresponding set in the form of a dataframe. This funtion creates a vector of lists of length 1 and fills the component data with the content of parameter data.

Value

As described above, input data in a format suitable for functions operating on multiset data collections.

Author(s)

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

See Also

checkSets

Examples

```
singleSetData = matrix(rnorm(100), 10,10);
encapsData = fixDataStructure(singleSetData);
length(encapsData)
names(encapsData[[1]])
dim(encapsData[[1]]$data)
all.equal(encapsData[[1]]$data, singleSetData);
```

---

labels2colors *Convert numerical labels to colors.*

Description

Converts a vector or array of numerical labels into a corresponding vector or array of colors corresponding to the labels.

Usage

```
labels2colors(labels, zeroIsGrey = TRUE, colorSeq = NULL)
```

Arguments

labels	Vector of non-negative integer labels.
zeroIsGrey	If TRUE, labels 0 will be assigned color grey. Otherwise, labels below 1 will trigger an error.
colorSeq	Color sequence corresponding to labels. If not given, a standard sequence will be used.

Details

The standard sequence start with well-distinguishable colors, and after about 40 turns into a quasi-random sampling of all colors available in R with the exception of all shades of grey (and gray).

If the input labels have a dimension attribute, it is copied into the output, meaning the dimensions of the returned value are the same as those of the input labels.

Value

A vector or array of character strings of the same length or dimensions as labels.

Author(s)

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

Examples

```
labels = c(0:20);
labels2colors(labels);
```

`mergeCloseModules` *Merge close modules of gene expression data.*

Description

Merges modules in gene expression networks that are too close as measured by the correlation of their eigengenes.

Usage

```
mergeCloseModules(exprData, colors,
                  cutHeight = 0.2,
                  MEs = NULL,
                  impute = TRUE,
                  useAbs = FALSE,
                  iterate = TRUE,
                  relabel = FALSE,
                  colorSeq = NULL,
                  getNewMEs = TRUE,
                  getNewUnassdME = TRUE,
                  useSets = NULL,
                  checkDataFormat = TRUE,
                  unassdColor = ifelse(is.numeric(colors), 0, "grey"),
                  trapErrors = FALSE,
                  verbose = 1, indent = 0)
```

Arguments

`exprData`	Expression data, either a single data frame with rows corresponding to samples and columns to genes, or in a multi-set format (see `checkSets`). See `checkDataStructure` below.
`colors`	A vector (numeric, character or a factor) giving module colors for genes. The method only makes sense when genes have the same color label in all sets, hence a single vector.
`cutHeight`	Maximum dissimilarity (i.e., 1-correlation) that qualifies modules for merging.
`MEs`	If module eigengenes have been calculated before, the user can save some computational time by inputting them. `MEs` should have the same format as `exprData`. If they are not given, they will be calculated.
`impute`	Should missing values be imputed in eigengene calculation? If imputation is disabled, the presence of `NA` entries will cause the eigengene calculation to fail and eigengenes will be replaced by their hubgene approximation. See `moduleEigengenes` for more details.

useAbs	Specifies whether absolute value of correlation or plain correlation (of module eigengenes) should be used in calculating module dissimilarity.
iterate	Controls whether the merging procedure should be repeated until there is no change. If FALSE, only one iteration will be executed.
relabel	Controls whether, after merging, color labels should be ordered by module size.
colorSeq	Color labels to be used for relabeling. Defaults to the standard color order used in this package if colors are not numeric, and to integers starting from 1 if colors is numeric.
getNewMEs	Controls whether module eigengenes of merged modules should be calculated and returned.
getNewUnassdME	When doing module eigengene manipulations, the function does not normally calculate the eigengene of the 'module' of unassigned ('grey') genes. Setting this option to TRUE will force the calculation of the unassigned eigengene in the returned newMEs, but not in the returned oldMEs.
useSets	A vector of scalar allowing the user to specify which sets will be used to calculate the consensus dissimilarity of module eigengenes. Defaults to all given sets.
checkDataFormat	If TRUE, the function will check exprData and MEs for correct multi-set structure. If single set data is given, it will be converted into a format usable for the function. If FALSE, incorrect structure of input data will trigger an error.
unassdColor	Specifies the string that labels unassigned genes. Module of this color will not enter the module eigengene clustering and will not be merged with other modules.
trapErrors	Controls whether computational errors in calculating module eigengenes, their dissimilarity, and merging trees should be trapped. If TRUE, errors will be trapped and the function will return the input colors. If FALSE, errors will cause the function to stop.
verbose	Controls verbosity of printed progress messages. 0 means silent, up to (about) 5 the verbosity gradually increases.
indent	A single non-negative integer controlling indentation of printed messages. 0 means no indentation, each unit above that adds two spaces.

Details

This function returns the color labels for modules that are obtained from the input modules by merging ones that are closely related. The relationships are quantified by correlations of module eigengenes; a "consensus" measure is defined as the minimum over the corresponding relationship in each set. Once the (dis-)similarity is calculated, average linkage hierarchical clustering of the module eigengenes is performed, the dendrogram is cut at the height cutHeight and modules on each branch are merged. The process is (optionally) repeated until no more modules are merged.

If, for a particular module, the module eigengene calculation fails, a hubgene approximation will be used.

The user should be aware that if a computational error occurs and trapErrors==TRUE, the returned list (see below) will not contain all of the components returned upon normal execution.

Value

If no errors occurred, a list with components

colors	Color labels for the genes corresponding to merged modules. The function attempts to mimic the mode of the input colors: if the input colors is numeric, character and factor, respectively, so is the output. Note, however, that if the fnction performs relabeling, a standard sequence of labels will be used: integers starting at 1 if the input colors is numeric, and a sequence of color labels otherwise (see colorSeq above).
dendro	Hierarchical clustering dendrogram (average linkage) of the eigengenes of the most recently computed tree. If iterate was set TRUE, this will be the dendrogram of the merged modules, otherwise it will be the dendrogram of the original modules.
oldDendro	Hierarchical clustering dendrogram (average linkage) of the eigengenes of the original modules.
cutHeight	The input cutHeight.
oldMEs	Module eigengenes of the original modules in the sets given by useSets.
newMEs	Module eigengenes of the merged modules in the sets given by useSets.
allOK	A boolean set to TRUE.

If an error occurred and trapErrors==TRUE, the list only contains these components:

colors	A copy of the input colors.
allOK	a boolean set to FALSE.

Author(s)

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

---

moduleColor.getMEprefix

Get the prefix used to label module eigengenes.

Description

Returns the currently used prefix used to label module eigengenes. When returning module eigengenes in a dataframe, names of the corresponding columns will start with the given prefix.

Usage

```
moduleColor.getMEprefix()
```

Details

Returns the prefix used to label module eigengenes. When returning module eigengenes in a dataframe, names of the corresponding columns will consist of the corresponfing color label preceded by the given prefix. For example, if the prefix is "PC" and the module is turquoise, the corresponding module eigengene will be labeled "PCturquoise". Most of old code assumes "PC", but "ME" is more instructive and used in some newer analyses.

Value

A character string.

Author(s)

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

See Also

moduleColor.setMEprefix, moduleEigengenes

---

moduleColor.revisionDate

Get the last revision date of the package.

Description

Returns the last revision date of the package.

Usage

```
moduleColor.revisionDate()
```

Value

A character string.

Author(s)

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

`moduleColor.setMEprefix` *Set the prefix used to label module eigengenes.*

Description

Sets the prefix used to label module eigengenes. When returning module eigengenes in a dataframe, names of the corresponding columns will start with the given prefix.

Usage

```
moduleColor.setMEprefix(prefix)
```

Arguments

`prefix`	A character string of length 2. Recommended values are "PC" (the default start-up value) and "ME".

Details

Sets the prefix used to label module eigengenes. When returning module eigengenes in a dataframe, names of the corresponding columns will consist of the corresponfing color label preceded by the given prefix. For example, if the prefix is "PC" and the module is turquoise, the corresponding module eigengene will be labeled "PCturquoise". Most of old code assumes "PC", but "ME" is more instructive and used in some newer analyses.

Value

None.

Author(s)

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

See Also

`moduleColor.getMEprefix`, `moduleEigengenes`

`moduleColor.version` *Get the version number of the package.*

Description

Returns the version number of the package.

Usage

```
moduleColor.version()
```

Value

A character string.

Author(s)

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

`moduleEigengenes` *Calculate module eigengenes.*

Description

Calculates module eigengenes (1st principal component) of modules in a given single dataset.

Usage

```
moduleEigengenes(expr,
                 colors,
                 impute = TRUE,
                 nPC = 1,
                 align = "along average",
                 excludeGrey = FALSE,
                 grey = ifelse(is.numeric(colors),  0, "grey"),
                 subHubs = TRUE,
                 trapErrors = FALSE,
                 returnValidOnly = trapErrors,
                 softPower = 6,
                 verbose = 0, indent = 0)
```

Arguments

expr	Expression data for a single set in the form of a data frame where rows are samples and columns are genes (probes).
colors	A vector of the same length as the number of probes in expr, giving module color for all probes (genes). Color "grey" is reserved for unassigned genes.
impute	If TRUE, expression data will be checked for the presence of NA entries and if the latter are present, numerical data will be imputed, using function impute.knn and probes from the same module as the missing datum. The function impute.knn uses a fixed random seed giving repeatable results.
nPC	Number of principal components and variance explained entries to be calculated. Note that only the first principal component is returned; the rest are used only for the calculation of proportion of variance explained. The number of returned variance explained entries is currently min(nPC, 10). If given nPC is greater than 10, a warning is issued.
align	Controls whether eigengenes, whose orientation is undetermined, should be aligned with average expression (align = "along average", the default) or left as they are (align = ""). Any other value will trigger an error.
excludeGrey	Should the improper module consisting of 'grey' genes be excluded from the eigengenes?
grey	Value of colors designating the improper module. Note that if colors is a factor of numbers, the default value will be incorrect.
subHubs	Controls whether hub genes should be substituted for missing eigengenes. If TRUE, each missing eigengene (i.e., eigengene whose calculation failed and the error was trapped) will be replaced by a weighted average of the most connected hub genes in the corresponding module. If this calculation fails, or if subHubs==FALSE, the value of trapErrors will determine whether the offending module will be removed or whether the function will issue an error and stop.
trapErrors	Controls handling of errors from that may arise when there are too many NA entries in expression data. If TRUE, errors from calling these functions will be trapped without abnormal exit. If FALSE, errors will cause the function to stop. Note, however, that subHubs takes precedence in the sense that if subHubs==TRUE and trapErrors==FALSE, an error will be issued only if both the principal component and the hubgene calculations have failed.
returnValidOnly	Boolean. Controls whether the returned data frame of module eigengenes contains columns corresponding only to modules whose eigengenes or hub genes could be calculated correctly (TRUE), or whether the data frame should have columns for each of the input color labels (FALSE).
softPower	The power used in soft-thresholding the adjacency matrix. Only used when the hubgene approximation is necessary because the principal component calculation failed. It must be non-negative. The default value should only be changed if there is a clear indication that it leads to incorrect results.
verbose	Controls verbosity of printed progress messages. 0 means silent, up to (about) 5 the verbosity gradually increases.
indent	A single non-negative integer controlling indentation of printed messages. 0 means no indentation, each unit above that adds two spaces.

Details

Module eigengene is defined as the first principal component of the expression matrix of the corresponding module. The calculation may fail if the expression data has too many missing entries. Handling of such errors is controlled by the arguments subHubs and trapErrors. If subHubs==TRUE, errors in principal component calculation will be trapped and a substitute calculation of hubgenes will be attempted. If this fails as well, behaviour depends on trapErrors: if TRUE, the offending module will be ignored and the return value will allow the user to remove the module from further analysis; if FALSE, the function will stop.

From the user's point of view, setting trapErrors=FALSE ensures that if the function returns normally, there will be a valid eigengene (principal component or hubgene) for each of the input colors. If the user sets trapErrors=TRUE, all calculational (but not input) errors will be trapped, but the user should check the output (see below) to make sure all modules have a valid returned eigengene.

While the principal component calculation can fail even on relatively sound data (it does not take all that many "well-placed" NA to torpedo the calculation), it takes many more irregularities in the data for the hubgene calculation to fail. In fact such a failure signals there likely is something seriously wrong with the data.

Value

A list with the following components:

eigengenes	Module eigengenes in a dataframe, with each column corresponding to one eigengene. The columns are named by the corresponding color with an "ME" prepended, e.g., MEturquoise etc. If returnValidOnly==FALSE, module eigengenes whose calculation failed have all components set to NA.
averageExpr	If align == "along average", a dataframe containing average normalized expression in each module. The columns are named by the corresponding color with an "AE" prepended, e.g., AEturquoise etc.
varExplained	A dataframe in which each column corresponds to a module, with the component varExplained[PC, module] giving the variance of module module explained by the principal component no. PC. The calculation is exact irrespective of the number of computed principal components. At most 10 variance explained values are recorded in this dataframe.
nPC	A copy of the input nPC.
validMEs	A boolean vector. Each component (corresponding to the columns in data) is TRUE if the corresponding eigengene is valid, and FALSE if it is invalid. Valid eigengenes include both principal components and their hubgene approximations. When returnValidOnly==FALSE, by definition all returned eigengenes are valid and the entries of validMEs are all TRUE.
validColors	A copy of the input colors with entries corresponding to invalid modules set to grey if given, otherwise 0 if colors is numeric and "grey" otherwise.
allOK	Boolean flag signalling whether all eigengenes have been calculated correctly, either as principal components or as the hubgene average approximation.
allPC	Boolean flag signalling whether all returned eigengenes are principal components.

isPC	Boolean vector. Each component (corresponding to the columns in eigengenes) is TRUE if the corresponding eigengene is the first principal component and FALSE if it is the hubgene approximation or is invalid.
isHub	Boolean vector. Each component (corresponding to the columns in eigengenes) is TRUE if the corresponding eigengene is the hubgene approximation and FALSE if it is the first principal component or is invalid.
validAEs	Boolean vector. Each component (corresponding to the columns in eigengenes) is TRUE if the corresponding module average expression is valid.
allAEOK	Boolean flag signalling whether all returned module average expressions contain valid data. Note that returnValidOnly==TRUE does not imply allAEOK==TRUE: some invalid average expressions may be returned if their corresponding eigengenes have been calculated correctly.

Author(s)

Steve Horvath <SHorvath@mednet.ucla.edu>, Peter Langfelder <Peter.Langfelder@gmail.com>

References

Zhang, B. and Horvath, S. (2005), "A General Framework for Weighted Gene Co-Expression Network Analysis", Statistical Applications in Genetics and Molecular Biology: Vol. 4: No. 1, Article 17

See Also

svd, impute.knn

---

moduleNumber *Fixed-height cut of a dendrogram.*

Description

Detects branches of on the input dendrogram by performing a fixed-height cut.

Usage

```
moduleNumber(dendro, cutHeight = 0.9, minSize = 50)
```

Arguments

dendro	a hierarchical clustering dendorgram such as one returned by hclust.
cutHeight	Maximum joining heights that will be considered.
minSize	Minimum cluster size.

Details

All contiguous branches below the height cutHeight that contain at least minSize objects are assigned unique positive numerical labels; all unassigned objects are assigned label 0.

Value

A vector of numerical labels giving the assigment of each object.

Note

The numerical labels may not be sequential. See `normalizeLabels` for a way to put the labels into a standard order.

Author(s)

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

See Also

`hclust`, `cutree`, `normalizeLabels`

`multiSetMEs` *Calculate module eigengenes.*

Description

Calculates module eigengenes for several sets.

Usage

```
multiSetMEs(exprData,
            colors,
            universalColors = NULL,
            useSets = NULL,
            useGenes = NULL,
            impute = TRUE,
            nPC = 1,
            align = "along average",
            excludeGrey = FALSE,
        grey = ifelse(is.null(universalColors), ifelse(is.numeric(colors), 0, "grey"),
                      ifelse(is.numeric(universalColors), 0, "grey")),
            subHubs = TRUE,
            trapErrors = FALSE,
            returnValidOnly = trapErrors,
            softPower = 6,
            verbose = 1, indent = 0)
```

Arguments

exprData	Expression data in a multi-set format (see checkSets). A vector of lists, with each list corresponding to one microarray dataset and expression data in the component data, that is expr[[set]]$data[sample, probe] is the expression of probe probe in sample sample in dataset set. The number of samples can be different between the sets, but the probes must be the same.
colors	A matrix of dimensions (number of probes, number of sets) giving the module assignment of each gene in each set. The color "grey" is interpreted as unassigned.
universalColors	Alternative specification of module assignment. A single vector of length (number of probes) giving the module assignment of each gene in all sets (that is the modules are common to all sets). If given, takes precedence over color.
useSets	If calculations are requested in (a) selected set(s) only, the set(s) can be specified here. Defaults to all sets.
useGenes	Can be used to restrict calculation to a subset of genes (the same subset in all sets). If given, validColors in the returned list will only contain colors for the genes specified in useGenes.
impute	Logical. If TRUE, expression data will be checked for the presence of NA entries and if the latter are present, numerical data will be imputed, using function impute.knn and probes from the same module as the missing datum. The function impute.knn uses a fixed random seed giving repeatable results.
nPC	Number of principal components to be calculated. If only eigengenes are needed, it is best to set it to 1 (default). If variance explained is needed as well, use value NULL. This will cause all principal components to be computed, which is slower.
align	Controls whether eigengenes, whose orientation is undetermined, should be aligned with average expression (align = "along average", the default) or left as they are (align = ""). Any other value will trigger an error.
excludeGrey	Should the improper module consisting of 'grey' genes be excluded from the eigengenes?
grey	Value of colors or universalColors (whichever applies) designating the improper module. Note that if the appropriate colors argument is a factor of numbers, the default value will be incorrect.
subHubs	Controls whether hub genes should be substituted for missing eigengenes. If TRUE, each missing eigengene (i.e., eigengene whose calculation failed and the error was trapped) will be replaced by a weighted average of the most connected hub genes in the corresponding module. If this calculation fails, or if subHubs==FALSE, the value of trapErrors will determine whether the offending module will be removed or whether the function will issue an error and stop.
trapErrors	Controls handling of errors from that may arise when there are too many NA entries in expression data. If TRUE, errors from calling these functions will be trapped without abnormal exit. If FALSE, errors will cause the function to stop. Note, however, that subHubs takes precedence in the sense that if subHubs==TRUE and trapErrors==FALSE, an error will be issued only if both the principal component and the hubgene calculations have failed.

`returnValidOnly`
: Boolean. Controls whether the returned data frames of module eigengenes contain columns corresponding only to modules whose eigengenes or hub genes could be calculated correctly in every set (`TRUE`), or whether the data frame should have columns for each of the input color labels (`FALSE`).

`softPower`
: The power used in soft-thresholding the adjacency matrix. Only used when the hubgene approximation is necessary because the principal component calculation failed. It must be non-negative. The default value should only be changed if there is a clear indication that it leads to incorrect results.

`verbose`
: Controls verbosity of printed progress messages. 0 means silent, up to (about) 5 the verbosity gradually increases.

`indent`
: A single non-negative integer controlling indentation of printed messages. 0 means no indentation, each unit above that adds two spaces.

Details

This function calls `moduleEigengenes` for each set in `exprData`.

Module eigengene is defined as the first principal component of the expression matrix of the corresponding module. The calculation may fail if the expression data has too many missing entries. Handling of such errors is controlled by the arguments `subHubs` and `trapErrors`. If `subHubs==TRUE`, errors in principal component calculation will be trapped and a substitute calculation of hubgenes will be attempted. If this fails as well, behaviour depends on `trapErrors`: if `TRUE`, the offending module will be ignored and the return value will allow the user to remove the module from further analysis; if `FALSE`, the function will stop. If `universalColors` is given, any offending module will be removed from all sets (see `validMEs` in return value below).

From the user's point of view, setting `trapErrors=FALSE` ensures that if the function returns normally, there will be a valid eigengene (principal component or hubgene) for each of the input colors. If the user sets `trapErrors=TRUE`, all calculational (but not input) errors will be trapped, but the user should check the output (see below) to make sure all modules have a valid returned eigengene.

While the principal component calculation can fail even on relatively sound data (it does not take all that many "well-placed" `NA` to torpedo the calculation), it takes many more irregularities in the data for the hubgene calculation to fail. In fact such a failure signals there likely is something seriously wrong with the data.

Value

A vector of lists similar in spirit to the input `exprData`. For each set there is a list with the following components:

`data`
: Module eigengenes in a data frame, with each column corresponding to one eigengene. The columns are named by the corresponding color with an `"ME"` prepended, e.g., `MEturquoise` etc. Note that, when `trapErrors == TRUE` and `returnValidOnly==FALSE`, this data frame also contains entries corresponding to removed modules, if any. (`validMEs` below indicates which eigengenes are valid and `allOK` whether all module eigengens were successfully calculated.)

`averageExpr`
: If `align == "along average"`, a dataframe containing average normalized expression in each module. The columns are named by the corresponding color with an `"AE"` prepended, e.g., `AEturquoise` etc.

varExplained — A dataframe in which each column corresponds to a module, with the component varExplained[PC, module] giving the variance of module module explained by the principal component no. PC. This is only accurate if all principal components have been computed (input nPC = NULL). At most 5 principal components are recorded in this dataframe.

nPC — A copy of the input nPC.

validMEs — A boolean vector. Each component (corresponding to the columns in data) is TRUE if the corresponding eigengene is valid, and FALSE if it is invalid. Valid eigengenes include both principal components and their hubgene approximations. When returnValidOnly==FALSE, by definition all returned eigengenes are valid and the entries of validMEs are all TRUE.

validColors — A copy of the input colors (universalColors if set, otherwise colors[, set]) with entries corresponding to invalid modules set to grey if given, otherwise 0 if the appropriate input colors are numeric and "grey" otherwise.

allOK — Boolean flag signalling whether all eigengenes have been calculated correctly, either as principal components or as the hubgene approximation. If universalColors is set, this flag signals whether all eigengenes are valid in all sets.

allPC — Boolean flag signalling whether all returned eigengenes are principal components. This flag (as well as the subsequent ones) is set independently for each set.

isPC — Boolean vector. Each component (corresponding to the columns in eigengenes) is TRUE if the corresponding eigengene is the first principal component and FALSE if it is the hubgene approximation or is invalid.

isHub — Boolean vector. Each component (corresponding to the columns in eigengenes) is TRUE if the corresponding eigengene is the hubgene approximation and FALSE if it is the first principal component or is invalid.

validAEs — Boolean vector. Each component (corresponding to the columns in eigengenes) is TRUE if the corresponding module average expression is valid.

allAEOK — Boolean flag signalling whether all returned module average expressions contain valid data. Note that returnValidOnly==TRUE does not imply allAEOK==TRUE: some invalid average expressions may be returned if their corresponding eigengenes have been calculated correctly.

Author(s)

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

See Also

moduleEigengenes

normalizeLabels *Transform numerical labels into normal order.*

Description

Transforms numerical labels into normal order, that is the largest group will be labeled 1, next largest 2 etc. Label 0 is optionally preserved.

Usage

```
normalizeLabels(labels, keepZero = TRUE)
```

Arguments

labels	Numerical labels.
keepZero	If TRUE (the default), labels 0 are preserved.

Value

A vector of the same length as input, containing the normalized labels.

Author(s)

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

orderMEs *Put close eigenvectors next to each other*

Description

Reorder given (eigen-)vectors such that similar ones (as measured by correlation) are next to each other.

Usage

```
orderMEs(MEs, greyLast = TRUE,
         greyName = paste(moduleColor.getMEprefix(), "grey", sep=""),
         orderBy = 1, order = NULL,
         useSets = NULL,  verbose = 0, indent = 0)
```

Arguments

MEs	Module eigengenes in a multi-set format (see checkSets). A vector of lists, with each list corresponding to one dataset and the module eigengenes in the component data, that is MEs[[set]]$data[sample, module] is the expression of the eigengene of module module in sample sample in dataset set. The number of samples can be different between the sets, but the modules must be the same.
greyLast	Normally the color grey is reserved for unassigned genes; hence the grey module is not a proper module and it is conventional to put it last. If this is not desired, set the parameter to FALSE.
greyName	Name of the grey module eigengene.
orderBy	Specifies the set by which the eigengenes are to be ordered (in all other sets as well). Defaults to the first set in useSets (or the first set, if useSets is not given).
order	Allows the user to specify a custom ordering.
useSets	Allows the user to specify for which sets the eigengene ordering is to be performed.
verbose	Controls verbostity of printed progress messages. 0 means silent, nonzero verbose.
indent	A single non-negative integer controling indentation of printed messages. 0 means no indentation, each unit above zero adds two spaces.

Details

Ordering module eigengenes is useful for plotting purposes. For this function the order can be specified explicitly, or a set can be given in which the correlations of the eigengenes will determine the order. For the latter, a hierarchical dendrogram is calculated and the order given by the dendrogram is used for the eigengenes in all other sets.

Value

A vector of lists of the same type as MEs containing the re-ordered eigengenes.

Author(s)

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

See Also

moduleEigengenes, multiSetMEs, consensusOrderMEs

plotHclustColors *Plot color rows corresponding to modules*

Description

Plot color bars corresponding to modules, usually beneath a dendrogram.

Usage

```
plotHclustColors(dendro, colors, rowLabels = NULL, cex.rowLabels = 0.9, ...)
```

Arguments

dendro	A dendrogram such as returned by hclust.
colors	Coloring of objects on the dendrogram. Either a vector (one color per object) or a matrix (can also be an array or a data frame) with each column giving one color per object. Each column will be plotted as a horizontal row of colors under the dendrogram.
rowLabels	Labels for the colorings given in colors. The labels will be printed to the left of the color rows in the plot. If the argument is given, it must be a vector of length equal to the number of columns in colors. If not given, names(colors) will be used if available. If not, sequential numbers starting from 1 will be used.
cex.rowLabels	Font size scale factor for the row labels. See par.
...	Other parameters to be passed on to the plotting method (such as main for the main title etc).

Details

It is often useful to plot module assignment (by color) that was obtained by cutting a hierarchical dendrogram, to visually check whether the obtained modules are meaningful, or which one of several possible module assignments looks best. One way to do it to section the screen into two parts, plot the dendrogram (via plot(hclust)) in the upper section and use this function to plot colors in the order corresponding to the dendrogram in the lower section.

Value

None.

Author(s)

Steve Horvath <SHorvath@mednet.ucla.edu> and Peter Langfelder <Peter.Langfelder@gmail.com>

See Also

cutreeDynamic for module detection in a dendrogram.

removeGreyME *Removes the grey eigengene from a given collection of eigengenes.*

Description

Given module eigengenes either in a single data frame or in a multi-set format, removes the grey eigengenes from each set. If the grey eigengenes are not found, a warning is issued.

Usage

```
removeGreyME(MEs, greyMEName = paste(moduleColor.getMEprefix(), "grey", sep=""))
```

Arguments

MEs	Module eigengenes, either in a single data frame (typicaly for a single set), or in a multi-set format. See checkSets for a description of the multi-set format.
greyMEName	Name of the module eigengene (in each corresponding data frame) that corresponds to the grey color. This will typically be "PCgrey" or "MEgrey". If the module eigengenes were calculated using standard functions in this library, the default should work.

Value

Module eigengenes in the same format as input (either a single data frame or a vector of lists) with the grey eigengene removed.

Author(s)

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

standardColors *Colors this library uses for labeling modules.*

Description

Returns the vector of color names in the order they are assigned by other functions in this library.

Usage

```
standardColors(n = NULL)
```

Arguments

n	Number of colors requested. If NULL, all (approx. 450) colors will be returned. Any other invalid argument such as less than one or more than maximum (length(standardColors())) will trigger an error.

Value

A vector of character color names of the requested length.

Author(s)

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

Examples

```
standardColors(10);
```

Index