maDB

April 19, 2009

`EexprSet-class`	*Class "EexprSet" an extended verision of the "exprSet" class from "Biobase"*

Description

The "EexprSet" class extends the "exprSet" class from the package "Biobase", additional functionality that has been included in this type of object are a "weights" slot, that can be used to mark flagged genes, some generic methods like "drawMA" and "drawHistogram" as well the functions "getM" and "getA" that are quite usefull. The "EexprSet" class can be used for both two color micro arrays and Affymetrix GeneChips. Limma objects can be converted with the `createEexprSetFromLimma` function

Warning

This object is deprecated and will be removed in the next version! Use MadbSet instead! MadbSet provides the same functionality and shares the same methods and functions with EexprSet.

Objects from the Class

Objects can be created by calls of the form `new("EexprSet", ...)`. if to this call a "exprSet" is submitted, this is converted to a "EexprSet" object. All of the following afguments can be passed also through the constructor call.

Slots

`type:` Object of class `"character"` specifying if the data origins from a one color array ("Affymetrix") or from a two color array ("TwoColor"). This argument is optional.

`printer:` Object of class `"Layout"` same as in the limma package the printer attribute. Defines the number of rows and columns per subgrid as well as the rows and columns of subgrids on the array.

`samples:` Object of class `"list"` a list containing the sample names. Optional, the names of the chips/arrays are anyway stored in the rownames of the "exprSet".

`array.pairs:` Object of class `"numeric"` a vector that specifies what columns belong to each other. like the red and green intensities per array. In Affymetrix chips this is a optional parameter, for two color micro arrys this parameter should be defined. e.g with "array.pairs=c(1,-1,2,-2)" for a "EexprSet" that contains data from two arrays (and has therefore 4 expression columns), the number defines what columns belong to each other, the signs define the color

(positive for Red, negative for Green, in the tradition of the M value that is `log2(R)-log2(G)`). By using the "createEexprSetFromLimma" one has not to care about all these erguments, as the function will read all this from the limma object.

weights: Object of class `"matrix"` a matrix with zeros (flagged features) and ones (not flagged ones). should always have the same number of columns and rows as the "exprMatrix". Optional argument.

exprs: Object of class `"exprMatrix"` the object that contains the expression information for each gene/feature. In Affymetrix GeneChip experiments it has the same number of columns as chips and the same number of rows as probe sets on each chip. For two color micro arrays it has 2xnumber of array columns (as the red and also the green channel are stored each into one column).

se.exprs: Object of class `"exprMatrix"` same as "exprs".

phenoData: Object of class `"phenoData"` look at the documentation of the "exprSet" object.

description: Object of class `"characterORMIAME"` look at the documentation of the "exprSet" object.

annotation: Object of class `"character"` look at the documentation of the "expSet" class.

notes: Object of class `"character"` look at the documentation of the "exprSet" class.

genes: Object of class `"matrix"`, holds information about the genes that belong to the rows in the `exprs` slot. This slot is automatically loaded with the `genes` slot of a **limma** object (MAList or RGList) when the function `toEexprSet` is used to transform those objects into a EexprSet object.

Extends

Class `"exprSet"`, directly.

Methods

average See documenation `average`.

drawBioControls `signature(object = "EexprSet")`: Draws the expression intensities from the biotynilated control sequences that Affymetrix included in their GeneChips.

initialize internal function called when a new `EexprSet` is created.

drawHistogram `(object,index=NULL,type=NULL,draw.pairs=FALSE,nr.breaks=250,draw.leg` Draws histograms of the intensities of the different chips. `index`: with index you can specify the channels (intensities) for which a histogram should be drawn. By default a histogram with all channels available will be drawn. `type`: not used yet... `draw.pairs`: not used yet... `nr.breaks`: the number of breaks for the histogram. `draw.legend`: if a legend should be drawn into the plot. `col`: the color in which the lines should be drawn. By default for each line an other color will be used.

drawMA `(object,array.pairs=NULL,g=NULL,r=NULL,log.values=NULL,one.image=TRUE,dra` Draws an M versus A plot of the data. By submitting r and g one can specify what chips (or channel) should be used as the red intensities and which as the green channel. `array.pairs`: a vector with the indizes of the channels to compare. Red signal channels should be specified as positive numbers, green channels as negative ones. When the object was created with the `createEexprSetFromLimmaClass` this argument is already stored in the object itself. Usually it is better to specify the channels to draw with the `r` and `g` arguments. `g`: the index of the green intensity channel in the `exprs` matrix. `r`: the same for the red signal channel. `log.values`: if the values in the object are already in log2 scale. This argument is optional as the function checks for log scale values anyway. `one.image`: if all MA plots should be

drawn into one image, only usefull when array.pairs are specified! `draw.flagged`: if the flagged genes should also be drawn into the plot (as orange points). `use.weights.from`: index of the channel from which the weights should be used; if for one MA plot only the weights of one of the two signal channels should be used, otherwise only those genes will be drawn, that are not flagged bad in both channels. `main`: the title for the plot. `colramp`: a function that takes numbers as input and returns colors (just like the heat.colors or topo.colors). If submitted the points are colored according to the density of spots in the place of the point. Requires the "geneplotter" package. `col`: the color or vector of colors in which the points should be drawn.

drawPolyAControls `signature(object = "EexprSet")`: Draws the expression intensities of the poly A control sequences from Affymetrix.

getA `(object,r,g,remove.flagged=FALSE)`: returns the A values (average intensity, A = 1/2 * log2(R*G)). To define which columns in the expression matrix represent the green and which the red channel one can submit the r and g parameters with the index of the red signals and green signals respectively. For example `getA(Eset,r=2,g=1)` uses the second column of the exprs slot of the Eset object as red signal channel in the formula above and the first column as green signal channel. It is also possible to submit a numerical vector with the indizes of the red signal channels as r (for example `r=c(2,4)`) and the same for the g parameter (`g=c(1,3)`. `This returns an A values matrix with two columns, the first with A values using the the 2nd column as red and 1st column as green channel, the second column of A values was calculated by using the 4th column as red and the 3rd as green channel. By setting the parameter \code{remove.flagged} to \code{TRUE} will give all genes that have a weight of 0 (in the \code{weights} slot of the \code{EexprSet} object) in one of the signal channels a \code{NA} value.`)

getArrays `(object,...)`: returns the names of the arrays of the `EexprSet` object submitted.

getM `(object,r,g,remove.flagged=FALSE)`: Same as the getA function, but this function returns the regulation M (M = log2(R/G)).

getSignalChannels `(object,...)`: returns the signal channel names of the `EexprSet` object submitted.

getWeights `object)`: Returns the weights for the single intensities. A weight of 0 means that the feature was flagged, a weight of 1 means that the signal is ok.

toExprSet `(object)`: Casts the EexprSet object in a expSet object.

loadFromDB `(object,connection,name,pk,v=TRUE)`: loads a formerly saved (with the function `publishToDB`) `EexprSet` object from the database. This method requires the **tkfiDB** package. Arguments: `object`: a instance of the `EexprSet` class, submit `new("EexprSet")` to create a new one. `connection`: the database connection to a PostgreSQL database. `name`: the experiment name (experiment title) that you want to load from the database. To get a overview of the available experiments in the database you can call `dbGetExperimentInfo(connection)`. `pk`: optional argument. Useful to load a subset of an experiment from the database. Simply submit the primary keys from the signal channels (column `signal_channels_pk` in the result from the `dbGetExperimentInfo` function). `v`: if TRUE additional informations will be printed to the console.

publishToDB `signature(object = "EexprSet")`: stores the object into a PostgreSQL database. Call `help(publishToDB)` to get more information of this method. This method requires the **tkfiDB** package.

writeTable `(object,file,mas5calls,include.annotation)`: writes the expression values to a tab delimited table. Arguments: `object`: the object from the type `EexprSet` that contains the normalized values. `file`: the file where the data should be written into.

`mas5calls`: optional, for MAS5 normalized chips. If submitted, the calls will be included in the output file.

Author(s)

Johannes Rainer

See Also

`createEexprSetFromLimma`, `average`

`Layout-class` *Microarray grid and print tip layout description object*

Description

This object can be used to describe the layout of a microarray (the number of sub grids or the number of spots per sub grid (print tip group)). The description of the array layout is essential for e.g. a print tip normalization method.

Slots

`n.grid.row`: The number of sub grid rows onto the array.

`n.grid.col`: The number of sub grid columns on the array.

`n.spot.row`: The number of spot columns per sub grid.

`n.spot.col`: The number of spot rows per sub grid.

Methods

No methods defined with class "Layout" in the signature.

Author(s)

Johannes Rainer

See Also

`MadbSet-class`

MAtoG *Calculate the green intensity values from the M and A values given.*

Description

MAtoG This function calculates the green intensity values if one has only M and A values.

Usage

```
MAtoG(M,A)
```

Arguments

M	A vector (or matrix) of M values
A	A vector (or matrix) of A values

Details

M values are defined by the formula `M = log2(R/G)` and the A values by `A = 1/2 * log2(R*G)`. This function simply calculates from the M and A values back to the G (green) intensity values by performing the following calculation: `sqrt((2^(2*A))/2^M)`.

Author(s)

Johannes Rainer

See Also

MAtoR

MAtoR *Calculate the red intensity values from the M and A values given.*

Description

MAtoR This function calculates the red intensity values if one has only M and A values.

Usage

```
MAtoR(M,A)
```

Arguments

M	A vector (or matrix) of M values
A	A vector (or matrix) of A values

Details

M values are defined by the formula `M = log2(R/G)` and the A values by `A = 1/2 * log2(R*G)`. This function simply calculates from the M and A values back to the R (red) intensity values.

Author(s)

Johannes Rainer

See Also

MAtoG

MadbSet-class *Class "MadbSet" an extended verision of the "ExpressionSet" class from "Biobase"*

Description

The "MadbSet" class directly extends the "ExpressionSet" class from the package "Biobase", additional functionality that has been included in this type of object are a "weights" slot, that can be used to mark flagged genes, some generic methods like "drawMA" and "drawHistogram" as well the functions "getM" and "getA" that are quite usefull. The "MadbSet" class can be used for both two color micro arrays and Affymetrix GeneChips. Limma objects (`RGList` or `MAList`) or `ExpressionSet` objects can be converted with the `newMadbSet` function

Objects from the Class

Objects can be created by calls of the form `new("MadbSet", ...)` or using the function `newMadbSet`. The use of the `newMadbSet` function is preferred, since it allows to convert other objects to a MadbSet object.

Slots

`type:` Object of class `"character"` specifying if the data origins from a one color array ("Affymetrix") or from a two color array ("TwoColor"). This argument is optional.

`printer:` Object of class `"Layout"` same as in the limma package the printer attribute. Defines the number of rows and columns per subgrid as well as the rows and columns of subgrids on the array.

`samples:` Object of class `"list"` a list containing the sample names. Optional parameter.

`array.pairs:` Object of class `"numeric"` a vector that specifies what columns belong to each other. like the red and green intensities per array. In Affymetrix chips this is a optional parameter, for two color micro arrys this parameter should be defined. e.g with "array.pairs=c(1,-1,2,-2)" for a "MadbSet" that contains data from two arrays (and has therefore 4 expression columns), the number defines what columns belong to each other, the signs define the color (positive for Red, negative for Green, in the tradition of the M value that is `log2(R)-log2(G)`).

`weights:` Object of class `"matrix"` a matrix with zeros (flagged features) and ones (not flagged ones). should always have the same number of columns and rows as the "exprMatrix". Optional argument.

`exprs:` Inherited from `ExpressionSet`, for more information refer to the `ExpressionSet` documentation.

`se.exprs:` Inherited from `ExpressionSet`, for more information refer to the `ExpressionSet` documentation.

`phenoData:` Inherited from `ExpressionSet`, for more information refer to the `ExpressionSet` documentation.

annotation: Inherited from ExpressionSet, for more information refer to the ExpressionSet documentation.

genes: Object of class "matrix", holds information about the genes that belong to the rows in the exprs slot. This slot is automatically loaded with the genes slot of a **limma** object (MAList or RGList) when the function newMadbSet is used to transform those objects into a MadbSet.

Extends

Class "ExpressionSet", directly.

Methods

average See documenation average.

drawHistogram (object,index=NULL,type=NULL,draw.pairs=FALSE,nr.breaks=250,draw.leg
Draws histograms of the intensities of the different chips. index: with index you can specify the channels (intensities) for which a histogram should be drawn. By default a histogram with all channels available will be drawn. type: not used yet... draw.pairs: not used yet... nr.breaks: the number of breaks for the histogram. draw.legend: if a legend should be drawn into the plot. col: the color in which the lines should be drawn. By default for each line an other color will be used.

drawMA (object,array.pairs=NULL,g=NULL,r=NULL,log.values=NULL,one.image=TRUE,dra
Draws an M versus A plot of the data. By submitting r and g one can specify what chips (or channel) should be used as the red intensities and which as the green channel. r and g can either be the index of the respective sample in the ExpressionMatrix (exprs slot), or the name of the sample (i.e. the Affymetrix cel file name). array.pairs: a vector with the indizes of the channels to compare. Red signal channels should be specified as positive numbers, green channels as negative ones. When the object was created with the newMadbSet this argument is already stored in the object itself. Usually it is better to specify the channels to draw with the r and g arguments. g: the index of the green intensity channel in the exprs matrix (or the name of the sample/array). r: the same for the red signal channel. log.values: if the values in the object are already in log2 scale. This argument is optional as the function checks for log scale values anyway. one.image: if all MA plots should be drawn into one image, only usefull when array.pairs are specified! draw.flagged: if the flagged genes should also be drawn into the plot (as orange points). use.weights.from: index of the channel from which the weights should be used; if for one MA plot only the weights of one of the two signal channels should be used, otherwise only those genes will be drawn, that are not flagged bad in both channels. main: the title for the plot. colramp: a function that takes numbers as input and returns colors (just like the heat.colors or topo.colors). If submitted, the points are colored according to the density of spots in the place of the point. Requires the **geneplotter** package. col: the color or vector of colors in which the points should be drawn.

getA (object,r,g,remove.flagged=FALSE): returns the A values (average intensity, A = 1/2 * log2(R*G)). To define which columns in the expression matrix represent the green and which the red channel one can submit the r and g parameters with the index of the red signals and green signals respectively (or the names of the columns of the samples in the exprs slot). For example getA(Eset,r=2,g=1) uses the second column of the exprs slot of the Eset object as red signal channel in the formula above and the first column as green signal channel. It is also possible to submit a numerical vector with the indizes of the red signal channels as r (for example r=c(2,4)) and the same for the g parameter (g=c(1,3). This returns an A values matrix with two columns, the first with A values using the the 2nd column as red and 1st column as green channel, the second column of A values was calculated by using the 4th column

as red and the 3rd as green channel. By setting the parameter \code{remove.
to \code{TRUE} will give all genes that have a weight of 0 (in the
\code{weights} slot of the \code{MadbSet} object) in one of the
signal channels a \code{NA} value.)

getArrays `(object,...)`: returns the names of the arrays of the `MadbSet` object submitted.

getM `(object,r,g,remove.flagged=FALSE)`: Same as the getA function, but this function returns the regulation (log2 ratio) values (M = log2(R/G)).

getSignalChannels `(object,...)`: returns the signal channel names of the `MadbSet` object submitted.

getWeights `object)`: Returns the weights for the single intensities. A weight of 0 means that the feature was flagged, a weight of 1 means that the signal is ok.

loadFromDB `(object,connection,name,pk,v=TRUE)`: loads a previously saved (with the function `publishToDB`) `MadbSet` object from the database. This method requires the **pgUtils** package. Arguments: `object`: a instance of the `MadbSet` class, submit `newMadbSet()` to create a new one. `connection`: the database connection to a PostgreSQL database. `name`: the experiment name (experiment title) that you want to load from the database. To get a overview of the available experiments in the database you can call `dbGetExperimentInfo(connection)`. `pk`: optional argument. Useful to load a subset of an experiment from the database. Simply submit the primary keys from the signal channels (column `signal_channels_pk` in the result from the `dbGetExperimentInfo` function). `v`: if TRUE additional informations will be printed to the console.

publishToDB `signature(object = "MadbSet")`: stores the object into a PostgreSQL database. Call `help(publishToDB)` to get more information of this method. This method requires the **pgUtils** package.

writeTable `(object,file,mas5calls,include.annotation)`: writes the expression values to a tab delimited table. Arguments: `object`: the object from the type `MadbSet` that contains the normalized values. `file`: the file where the data should be written into. `mas5calls`: optional, for MAS5 normalized chips. If submitted, the calls will be included in the output file.

Note

For further informations and examples refer to the package vignette (documentation), which can be opened using `openMadbVignette`.

Author(s)

Johannes Rainer

See Also

`average`

`Sample-class` *Class "Sample"*

Description

In this object information about the sample that was hybridized onto a micro array can be stored into.

Slots

`name`: The name of the sample.

`genus`: The genus of the sample.

`species`: The species.

`line`: The cell line from wich the sample origins.

`individual`: The patient ID if the sample was extracted from a patient.

`sample.provider`: The sample provider.

`sex`: Sample sex.

`age`: Age (ov the individual).

`developmental.state`: The developmental state.

`tissue`: The tissue the sample comes from.

`celltype`: Sample celltype.

`genetic.modifications`: Genetic modifications of the sample.

`disease.state`: Disease state.

`ref.to.clinical.data`: References to clinical data.

`growth.conditions`: Sample growth conditions.

`in.vivo.treatments`: Sample in vivo treatments.

`tissue.harvesting.method`: Tisuue harvesting method.

`extract.method`: Sample extraction method.

`molecule.type`: Molecule type.

`amplification`: Samle amplification.

`amount`: Sample amount.

`treatment`: Sample treatment.

`exposure.time`: Sample treatment exposure time.

`description`: Sample description.

Methods

createNewDBTable `signature(object = "Sample")`: creates a new database table in wich objects of the type `sample` can be stored into.

getContentMatrix `signature(object = "Sample")`: returns the content of the individual slots of the `sample` object as a character matrix.

getPK `signature(object = "Sample")`: checks if a sample with the same attributes exists already in the database and returns its primary key or it inserts the sample in the database and returns the primary key of the newly inserted sample.

loadFromDB `signature(object = "Sample")`: loads the slots of the `sample` object with the attributes stored in the database.

Author(s)

Johannes Rainer

See Also

MadbSet-class loadFromDB publishToDB

SignalChannel-class

Microarray signal channel description

Description

With this object it is possible to describe the signal channels of a microarray experiment. This object establishes the linkage between the sample that was hybridized onto a specific signal channel of a microarray (signal channels correspond to the columns of the exprs slot of a MadbSet (or ExpressionSet)). \ For further informations and examples refer to the package vignette (which can be opened using the openMadbVignette).

Slots

hyb.date: The date when the sample was hybridized onto the array.

label.date: The date when the sample was labeled.

label.incorporation: The method that was used to label the RNA.

color: A description of the color that was used.

description: A description.

array.index: The index of the corresponding array in the MadbSet object.

sample.index: The index of the corresponding sample in the samples slot of the MadbSet object.

array.name: The micro array name or ID.

Methods

createNewDBTable signature(object = "SignalChannel"): creates a new database table that can be used to store objects from the type SignalChannel into.

getContentMatrix signature(object = "SignalChannel"): returns the content of the slots of the SignalChannel object as a character matrix.

show signature(object = "SignalChannel"): Displays the objects content.

Author(s)

Johannes Rainer

See Also

MadbSet-class, Sample-class, loadFromDB, publishToDB, getSignalChannels getArrays

`Similarity-class` *Class "Similarity"*

Description

This class holds information about the similarity between a vector and another one (or the rows or columns of a numerical matrix). An instance of this class will be returned as the result of the `dbSearchSimilarPattern` function. This object holds then information about the similarity (calculated with one of the different distance measurement functions (pearson, euclidian, spearman)) between a template gene pattern and a set of other gene expression or regulation patterns.

Objects from the Class

The function `dbSearchSimilarPattern` returns an instance of this class.

Slots

`distances`: Object of class `"numeric"`, this vector represents the distance (similarity) between numerical vectors.

`distance.metric`: Object of class `"character"`, the distance metric used to calculate the similarity.

`data`: Object of class `"matrix"`, the data matrix containing the values with which the template was compared.

`template`: Object of class `"numeric"`, the template vector.

`template.id`: Object of class `"character"`, the id of the template (if no custom template was used).

Methods

getData `signature(object = "Similarity")`: returns the data matrix.

getDistances `signature(object = "Similarity")`: returns the distances.

plot `signature(x = "Similarity")`: plots the template and the 5 top matching vectors (see `plotSimilarity` for more options).

print `signature(x = "Similarity")`: generic print function for the class Similarity.

sort `signature(x = "Similarity")`: sorts the data ascending or descending according to the distance.

Author(s)

Johannes Rainer

See Also

`dbSearchSimilarPattern` `plotSimilarity`

arrowProgress *Progress bar with an arrow*

Description

progress displays a simple progress bar with an arrow.

Usage

```
arrowProgress(percent,steps=20,label=TRUE)
```

Arguments

percent	The progress of an action in percent.
steps	The number of steps of the progress bar. Setting the steps to 20 means that each 5 percent to progress bar increases.
label	if the actual percentage should be written as a number too.

Details

This function can be used to display the progress of functions that may take long time.

Author(s)

Johannes Rainer

See Also

progress

Examples

```
arrowProgress(percent=70,steps=40)
```

average *Average signal channels of a microarray experiment*

Description

average allows to average replicated signal channels (or replicated features within a signal channel) of a MadbSet object.

Usage

```
average(object,average.which,log.scale=TRUE,method,array.names=NULL,v=TRUE,avera
```

Arguments

object	The MadbSet that contains the signal channels that should be averaged.
average.which	A numerical vector defining which columns of the exprs (which signal channels) slot of the object should be averaged. e.g. if the exprs slot contains 4 column and the columns 1 and 3 respectiveley 2 and 4 contain the expression values of replicated samples, c(1,2,1,2) has to be submitted to average the replicated samples.
log.scale	If the expression values should be log2 transformed before averaging.
method	The averaging method (one of c("mean","median","tukey.biweight")).
array.names	The names for the averaged signal channels (the colnames of the exprs slot in the returned EexprSet object).
v	Display some verbose messages during the execution of the function.
average.genes	If replicated spots / genes per signal channel should also be averaged (must have the same ID!).
exclude.flagged	If features with a weight of 0 (in the weights slot, means feature was flagged bad by the scanning software) should not be used to calculate the average across replicates. Could be problematic when doing a dye swap normalization with only two arrays with this function and some features are flagged bad on one and not flagged bad on the other array (function will use only the value from the not flagged feature in this case). In this case it could be better submitting exclude.flagged=FALSE.
only.good.spots	If only those features should be averaged across replicated arrays, that are not flagged bad in all arrays (Only for averaging accross arrays, not averaging within arrays!).
...	Additional parameters for a optional filter function.

Details

This function allows to average measured intensities of replicated microarrays (technical replicates). If the weights slot of the MadbSet object is not NULL and the exclude.flagged parameter is set TRUE, all genes that have weight of 0 (e.g. were flagged as bad spots by the scanning software) are excluded from the calculation of the average expression values (see description of the attribute exclude.flagged for more information).

Value

A MadbSet object that contains the averaged expression values for replicated microarrays.

Author(s)

Johannes Rainer

See Also

MadbSet-class

`calculateSimilarity` *Calculate similarities (distances) between a vector and the rows (columns) of a matrix*

Description

`calculateSimilarity` calculates the similarity (distance) between the submitted numeric vector and all rows respectively columns of a numerical matrix using the specified similarity (distance) measurement method.

Usage

```
calculateSimilarity(vector,matrix,orientation="h",distance.fun=distance.euclidia
```

Arguments

`vector`	The numerical vector for which the similarities should be calculated.
`matrix`	Numerical matrix. Similarities are calculated for the input vector (`vector` parameter) with all rows (or columns, depending on the `orientation` parameter) of this matrix.
`orientation`	If the vector should be compared with the rows (`"h"`) or columns (`"v"`) of the matrix.
`distance.fun`	Function to calculate the similarity.
`include.values`	Include input values in the result. If FALSE only a vector with the distances will be returned.

Details

This function is a simple and quick way to calculate similarities of a numerical vector and the columns (or rows) of a numerical matrix. `calculateSimilarity` can be used for example to search for genes with a common (similar) expression or regulation pattern than a input gene.

Author(s)

Johannes Rainer

See Also

`distance.pearson` `distance.euclidian` `distance.spearman` `dbSearchSimilarPattern`

Examples

```
In <- runif(5)
Testset <- rbind(a=runif(5,min=-1,max=2),b=runif(5,min=0,max=2),c=runif(5))
In
Testset
calculateSimilarity(vector=In,matrix=Testset,orientation="h")
```

checkForDifferentAnnotation
Detect gene annotation differences

Description

checkForDifferentAnnotation compares the gene annotation in the annotation database table with the gene annotation provided by the specified R annotation package. The IDs which differ in the annotation are returned as result.

Usage

```
checkForDifferentAnnotation(con,chip,check.columns=c("GenBank","UniGene","LocusL
```

Arguments

con	The connection object to the database (Connections can be opened with the dbConnect function from the **RdbiPgSQL** package).
chip	The chip for which the annotation should be checked. At the moment only Affymetrix chips are supported (eg. hgu133plus2).
check.columns	The identifiers that should be compared for the genes (Affymetrix probe sets).
v	if some additional informations should be written to the console.

Details

This function compares the annotation for the genes (or Affymetrix probe set IDs) that are available in a database table called annotation and the corresponding R annotation package (like the hgu133plus2 package). The annotation IDs specified by the check.columns attribute from the two different sources are compared for each gene and if they differ the gene and the different annotations are returned.

Author(s)

Johannes Rainer

See Also

dbUpdateAnnotation dbGetAnnotation

createDirs *Create directories*

Description

createDirs This function creates directories.

Usage

```
createDirs(dir)
```

Arguments

dir	The directories. As file delimiter, also in Windows, the Unix file delimiter "/" should be used!

Details

This function creates directories and subdirectories. If for example as dir the string "doc/tables/" is submited, the function creates the directory "doc", then the directory "tables" into the "doc" directory.

Author(s)

Johannes Rainer

See Also

createFile

createEexprSetFromLimma *Changing a RGList or MAList into a object of the type EexprSet*

Description

createEexprSetFromLimma This function creates a new object of the type EexprSet from a data object of the limma package. So the standard methods that are available for the EexprSet object are also available to tihs type of data.

Usage

```
createEexprSetFromLimma(object)
```

Arguments

object	The object (of the type RGList or MAList object from limma) to be converted.

Details

For two color micro array data, like the data objects in limma represent, the array.pairs attribute in the EexprSet object will contain information about what column correspondes to the red intensities and green intensities respectiveley of which array. So by calling for example the drawMA method the right intensities will be used to create the MA plot. Weights will also be set in the EexprSet object, as well as the printer layout if set in the limma object. The only thing that will be missing by now is the annotation (the $genes) object. This should be fixed soon...

warning

The EexprSet-class is deprecated! Use MadbSet instead! This function is replaced by the newMadbSet function.

Author(s)

Johannes Rainer

See Also

EexprSet-class, drawMA MadbSet newMadbSet

---

createFile *Create a file and all needed directories*

Description

createFile This function creates a new file just like the file.create function from R, but if needed it creates also directories.

Usage

```
createFile(file)
```

Arguments

file	The file name.

Details

This function creates a file also in a subdirectory, and creates this directory also if needed. If for example as file the string "doc/tables/testtable.txt" is submited, the function creates the directory "doc", then the directory "tables" into the "doc" directory and finally the file "testtable.txt" in this last sub directory. As file delimiter the "/" has always to be used, also in Windows!

Author(s)

Johannes Rainer

See Also

createDirs

`createJOINQuery` *Create a SQL query that joins two or more tables*

Description

This function creates SQL query strings that allow to join all available tables from the `maDB` database.

Usage

```
createJOINQuery(join, with, select = "*", r = TRUE)
```

Arguments

`join`	One of the two database tables that should be joined.
`with`	The database table with which the first one should be joined.
`select`	The attributes (column names) of the tables that should be returned. By default every column will be returned.
`r`	Attribute for joining the `comparisons` table with any other table. If `r=TRUE` the entries of the `comparisons` table will be joined with the other table using the information for the red signal channel, if `r=FALSE` the informtion for the green signal channel will be used.

Details

This function eases the usage of the various database tables of a `maDB` database. The list of tables that can be joined are: `comparisons, signal_channels, arrays, samples and experiments`.

Value

A string that can be used in the `dbSendQuery` (package RdbiPgSQL) function to retrieve the data.

Author(s)

Johannes Rainer

See Also

`dbGetComparisons`

Examples

```
##
## join the comparison table with the samples table
createJOINQuery(join="comparisons",with="samples")
```

createSampleFromTable *Create a new Sample object using data from a matrix*

Description

createSampleFromTable This function creates a new instance from the class Sample using the data in the matrix submitted as attribute. The column names of the matrix should correspond to the attribute names (slot names) of the Sample class. All "_" in the column names are replaced by a dot (for example for the slot exposure.time of the Sample class the table should have a column named "exposure_time").

Usage

```
createSampleFromTable(table)
```

Arguments

table	The data frame/matrix that contains the values with which the new Sample object should be filled.

Details

This function a new Sample object filling its slots with the values of the data table submitted.

Author(s)

Johannes Rainer

See Also

Sample-class

---

dbCalculateRegulations *Calculate regulation values from expression values in the database*

Description

dbCalculateRegulations calculates regulation values (M values, log2 fold change values) from expression values retrieved from the database and inserts these regulation values in the according database table (database tables *comparisons* and *regulation_values*, see the vignette for more information (openMadbVignette)).

Usage

```
dbCalculateRegulations(Con,exp.title,comparisons,v=TRUE)
```

Arguments

Con	The connection to the database (see dbConnect from the **RdbiPgSQL** package).
exp.title	The tile of the microarray experiment (data set) for which the regulation values should be calculated. Note that it is only possible to calculate regulation values for comparisons of arrays within a microarray experiment, which means that this regulation values can only be calculated for two color arrays or Affymetrix GeneChips which were normalized together.
comparisons	A list of comparisons for which M and A values should be calculated. To calculate regulation values for the first three chips of an experiment (comparing the expression values of the features/probe sets of the second signal channel (array) with those of the first (control) and of the third chip to those of the first) submit list(c(2,1),c(3,1)). Use the function dbGetExperimentInfo to get information of the arrays/signal channels of a experiment and the according index of the signal channels in this experiment.
v	If TRUE the progress of the calculation is shown using a progress bar.

Details

This function allows to calculate regulation values comparing expression values from two signal channels of a microarray experiment (dataset) in the maDB database. The according comparisons will be inserted into a database table called *comparisons* whichs attribute *r_signal_channel* links to the signal channel that was used as sample and whichs attribute *g_signal_channel* links to the signal channel that was used as control sample. The regulation values are calculated using the formula M=r-g and A=0.5*(r+g) where r and g are the expression values in log2 scale. The regulation values (M and A values) are inserted into a database table called *regulation_values*. Regulation values can be fetched from the database using the getMDB function. The function dbGetComparisons can be used to get informations which samples were compared in the comparisons in the database.

Author(s)

Johannes Rainer

See Also

dbGetExperimentInfo dbGetComparisons

---

dbGetComparisons *Get a list of all comparisons from the database*

Description

This function can be used to get all available information about comparisons that are stored in the database (i.e. which samples/signal channels are compared in which comparison).

Usage

```
dbGetComparisons(con, exp.title = NULL, join.tables = c("signal_channels", "samp
```

Arguments

con	The connection to the database.
exp.title	Get the comparisons that were created within a specific micro array experiment that was saved in the database. If not submitted all comparisons will be returned. To get the information about the experiments stored in the database, call dbGetExperimentInfo.
join.tables	The tables with which the comparisons will be joined. In the default case (which means join to all tables) all information concerning the comparisons (which experiment, sample, array ... was used) will be returned.

Value

The function returns a table with rows corresponding to the comparisons and columns to the attributes of the different tables (column names ending in "red" belong to the red signal channel and column names ending in "green" to the green respectively).

Note

For further information and examples refer to the package vignette (which can be opened using openMadbVignette).

Author(s)

Johannes Rainer

See Also

dbGetExperimentInfo, createJOINQuery

---

dbGetGeneRegulationRanking

Rank genes according to the number of comparisons they are regulated.

Description

This function can be used to generate a ranking list for genes that contains the number of comparisons (from a selected set of comparisons in the database) the gene was regulated (where regulation can be for example be defined as an M value bigger than 1 or smaller -1). The ranking can be restricted to a subset of genes and comparisons. Note that the comparisons have to be created in the database (from expresison values using the dbCalculateRegulations method). By default the function maps first the submitted ids to UniGene IDs and calcultates therefore the ranking for genes (not for, in the case of Affymetrix, probe sets), bevertheless by setting no.mapping.to.ug=TRUE this mapping to UniGene IDs will be skipped and the ranking will be performed id per id.

Usage

```
dbGetGeneRegulationRanking(con, ug = NULL, id = NULL, include.by.name = NULL, ex
```

Arguments

con	The connection to the (maDB) database.
ug	A set of UniGene identifiers. The annotation table has to be created if this option is used!
id	A set of ids (eg Affymetrix probe set IDs or the IDs that were used as rownames in the `EexprSet@exprs` tables that were inserted into the database).
include.by.name	The comparisons are joined to the samples table. So comparisons can be selected or using the `include.by.pk` attribute, or using the sample names of the samples that were used as *red* signal channels in the calculation of the regulation value (`M = log2(R / G)`).
exclude.by.name	Samples that should be excluded.
include.by.pk	The primary keys of the comparisons that should be included to create the ranking.
exclude.by.pk	The primary keys of the comparisons that should be excluded (makes only sense when the `include.by.name` was used, and not all of the comparisons that share the same sample names should be included).
count.cols	The entries (of the samples table) that should be used to count the regulation. For example if you have in your database samples with 6 hours, 8 hours and 24 hours exposure time and you want to count for each gene the number of comparisons with 6 and 8 hours where they were regulated and additionally the number of comparisons with 24 hours exposure time samples, submit `count.cols=list(c("6h","8h"),c("24h"))` and for the attribute search.col `search.col="exposure_time"`.
write.info.file	If true a file called "HL" with the current date and "Info.txt" will be saved in the current working directory which contains the information which comaprisons were used to generate the ranking.
search.column	The column of the samples table that should be used to distinguish between different comparisons (see attribute count.cols).
no.mapping.to.ug	If TRUE the ranking will be computed for each submitted id, if FALSE each id will be mapped to UniGene IDs, and those will then be used to count the regulations. Because of this mapping it is important, that the annotation table in the database holds the correct information (see `dbUpdateAnnotation` for more information).
m.up	Genes that have an M value bigger than this value will be counted as regulated (up–regulated!), the default is 1.
m.down	Genes that have an M value smaller than this value will be counted as down–regulated.
v	If additional information (progress bar...) should be written to the console.
chip	The chip or array type (eg hgu133plus2).

Details

coming soon...

Value

A table with the information about how many times a gene was regulated.

Author(s)

Johannes Rainer

See Also

dbUpdateAnnotation, publishToDB, dbCalculateRegulations

---

dbSearchSimilarPattern

Search for similar gene expression or regulation patterns in the database

---

Description

dbSearchSimilarPattern calculates the similarity of a template gene expression or regulation pattern to all gene expression (regulation) patterns in the maDB database.

Usage

```
dbSearchSimilarPattern(con,template=NULL,id=NULL,compare.ids=NULL,include.by.pk=
```

Arguments

con	The database connection.
template	A expression (regulation) pattern template (if NULL the id of the gene which expression (regulation) pattern should be used as template must be submitted with the id attribute).
id	The ID of the gene that should be used as template (for Affymetrix data the probe set id).
compare.ids	A set of genes with which the template should be compared. If not submitted all genes in the database will be used.
include.by.pk	The primary keys of the signal channels (or comparisons) that should be used to build the pattern.
include.by.name	The name of the samples that should be used to build the pattern.
exclude.by.pk	The primary keys of the signal channels (or comparisons) that should be excluded.
exclude.by.name	The name of the samples that should be excluded.
what	If similar expression pattern ("E") or regulation pattern ("M") should be searched.
include.values	See calculateSimilarity. If TRUE the returned Similarity class will contain also the data and makes it therefore possible to draw the data.

chip	The chip (array).
v	If additional informations should be plotted to the console.
distance.fun	The function to calculate similarities between vectors.
orientation	If genes should be compared ("h") or samples ("v").
column.names	The columns from the sample that should be used to build the column names of the data table. By default the column names of the result consist of the sample name, the sample tissue and the sample exposure time.
order	If the data columns should be ordered in the order of the primary keys (or names) submitted.

Details

This function calculates similarities between a template gene expression (regulation) pattern and all available patterns in the database. The function loads the values to compare from the database and calulates the similarity between the template and the matrix using the calculateSimilarity function. The user can specify the columns of the matrix by the include.by and exclude.by attributes and the rows by the compare.ids attribute (if not specified all genes will be used). If the user wants to compare a specific gene expression pattern (a custom pattern or a pattern of a gene in the database) with the expression pattern of all genes in the database, he has to specify the samples (signal channels) from which the expression values should be taken using the primary keys of the corresponding signal channels in the \code{signal_channels} database table, or by submitting the names of the samples that are hybridized onto the signal channels. For regulation values the primary keys of the comparisons in the \code{comparisons} table have to be used. As result an instance of the \code{Similarity} class will be returned (see \code{calculateSimilarity} for more information).\An example is shown in the package vignette (use openMadbVignette to open the documentation).

Value

A object from the type Similarity.

Author(s)

Johannes Rainer

See Also

distance.pearson distance.euclidian distance.spearman calculateSimilarity Similarity-class

---

deleteExperimentFromDB

Deleting a experiment from a database.

Description

deleteExperimentFromDB deletes an experiment (dataset) from the maDB database.

Usage

```
deleteExperimentFromDB(con,exp.title,v=TRUE)
```

Arguments

con	The connection to the database (returned by the dbConnect function of the **RdbiPgSQL** package).
exp.title	The name (title) from the experiment in the database that should be deleted (to get a list of available experiments in the database use dbGetExperimentInfo).
v	If TRUE additional information will be printed to the console.

Details

Like every function from the **pgUtils** and **maDB** packages that writes or updates values in a PostgreSQL database, this function uses transactions. The advantage of tranactions is, that the database status before running the function is automatically recreated if an error occurs during the function execution, or if the user aborts the execution.

Author(s)

Johannes Rainer

See Also

publishToDB dbGetExperimentInfo

---

distance.euclidian *Calculate vector similarity (distance) using euclidian distance metrics*

Description

distance.euclidian calculates the euclidian distance between two n–dimensional vectors.

Usage

```
distance.euclidian(x,y,na.rm=FALSE,...)
```

Arguments

x	The first vector.
y	The second vector.
na.rm	If NA values should be removed before any calculations take place.
...	Additional parameters. Not yet used.

Details

This simple function calculates the euclidian distance between two vectors using pythagoras theorem.

Author(s)

Johannes Rainer

See Also

distance.pearson distance.spearman dbSearchSimilarPattern

---

distance.pearson *Calculate vector similarity (distance) using pearsons correlation method*

Description

distance.pearson calculates the pearson correlation coefficient between two n–dimensional vectors.

Usage

```
distance.pearson(x,y,na.rm=FALSE,...)
```

Arguments

x	The first vector.
y	The second vector.
na.rm	If NA values should be removed before any calculations take place.
...	Additional parameters. Not yet used.

Details

This simple function calculates the correlation coefficient between two vectors.

Author(s)

Johannes Rainer

See Also

distance.euclidian distance.spearman dbSearchSimilarPattern

`distance.pearson.uncentered` *Calculate vector similarity (distance) using (uncentered) pearsons correlation method*

Description

`distance.pearson.uncentered` calculates the uncentered pearson correlation coefficient between two n–dimensional vectors.

Usage

```
distance.pearson.uncentered(x,y,na.rm=FALSE,...)
```

Arguments

`x`	The first vector.
`y`	The second vector.
`na.rm`	If `NA` values should be removed before any calculations take place.
`...`	Additional parameters. Not yet used.

Details

This simple function calculates the uncenetered correlation coefficient between two vectors.

Author(s)

Johannes Rainer

See Also

`distance.euclidian` `distance.spearman` `dbSearchSimilarPattern`

`distance.spearman` *Calculate spearmans rank correlation between two vectors*

Description

`distance.spearman` calculates spearmans rank correlation coefficient between two n–dimensional vectors.

Usage

```
distance.spearman(x,y,na.rm=FALSE,...)
```

Arguments

x	The first vector.
y	The second vector.
na.rm	If NA values should be removed before any calculations take place.
...	Additional parameters. Not yet used.

Details

This simple function calculates spearmans rank correlation coefficient between two vectors.

Author(s)

Johannes Rainer

See Also

distance.euclidian distance.pearson dbSearchSimilarPattern

---

drawMAPlot *Draw a standard MA plot*

Description

This function draws a simple M versus A plot with a lowess curve, a line for the mean A value, mean M value (drawn as red lines) and median M and A value (drawn as green line).

Usage

```
drawMAPlot(M, A, lwd = 1, ...)
```

Arguments

M	A vector with M values.
A	A vector with A values.
lwd	The line width.
...	Additional attributes for the plot call.

Value

An MA plot.

Author(s)

Johannes Rainer

See Also

drawMA

drawVolcanoPlot *Draw a volcano plot*

Description

drawVolcanoPlot plots a volcano plot scattering the M values (log2 ratio) on the x axis against the p value (-log10 the p value).

Usage

```
drawVolcanoPlot(M,p,m=1,p.cut=0.05,p.transform=log10,ylab=NULL,colramp=NULL,na.r
```

Arguments

M	The M values.
p	The test statistics (usually p values).
m	Where vertical lines should be drawn. By default two horizontal lines (at M=1 and M=-1) are added to the plot. Submit m=NA if none should be drawn.
p.cut	For what p value a horizontal line should be added to the plot (by default for a p value of p=0.05). Specify p.cut=NA if no line should be drawn.
p.transform	The function that should be applied to the p values. By default -log10. Note a − is added to each function submitted here.
ylab	The label for the y axis. By default -log10(p)
colramp	A color ramp (set of colors) that should be used to code the local point desity. This attribute requires the package geneplotter as the densCols function is used to calculate the local point density.
na.rm	If NA values should be removed.
...	Additional parameters for the plot function.

Details

This function plots M values against test statistics. A horizontal line for a specific p value (by default 0.05) is added as well as vertical lines for a specific M value (by default 1 and -1). The data points can be colored according to the local point density by specifying a specific color ramp (e.g. colramp=topo.colors).

Author(s)

Johannes Rainer

format.latex *Formats strings*

Description

format.latex Substitutes special characters in strings.

Usage

```
format.latex(x,...)
```

Arguments

x	The string that should be formatted.
...	not used yet.

Details

The function replaces all "$" and "%" and all other special characters, thus formatting the string properly for LaTeX.

Author(s)

Johannes Rainer

getAnnotation *Annotate a gene*

Description

This function retrieves annotation for a gene (or a list of genes) to different public repository IDs.

Usage

```
getAnnotation(id = NULL, con = NULL, method = "local", chip = NULL, v = TRUE, ..
```

Arguments

id	The gene ID (or IDs), for example Affymetrix probe set IDs, GenBank IDs...
con	The database connection to the "maDB" database, if method is "db". If the annotation should be done using the ensembl mart database using the **biomaRt** package the mart connection object should be submitted with con. If local annotation (for example using Bioconductors Annotation packages) should be used this atribute can be omitted.
method	The annotation method, one of c("local","db","mart").
chip	An identifier for the chip or package that should be used to annotate the id. e.g. "hgu133plus2" should be submitted for the Affymetrix HG-U133plus2 chip.
v	If additional output should be written to the console.
...	additional parameters.

Value

A matrix where the rows correspond to the ids (genes) and the columns to the available annotations for the genes.

Warning

The local annotation using annotation packages is not implemented yet!

Author(s)

Johannes Rainer

See Also

dbGetAnnotation

---

dbGetAnnotation *Annotate a list of IDs using the annotation table in the database.*

Description

dbGetAnnotation returns the annotation of the submitted IDs to available annotations in the maDB annotation database table. For an example refer to the package vignette (which can be opened using the openMadbVignette function).

Usage

```
dbGetAnnotation(Con,id,columns=c("gen_bank","description","uni_gene","locuslink"
```

Arguments

Con	The connection to the maDB database (created using the dbConnect function from the **RdbiPgSQL** package).
id	The IDs of the genes that should be annotated.
columns	the identifier to which the submitted ids should be annotated.
chip	the chip (or for two color arrays a oligo set identifier) from which the ids were taken from (e.g. hgu133plus2 for the Affymetrix GeneChip 133Plus2).
search.col	In wich of the columns the ids submitted with the id parameter should be searched for.
v	If TRUE additional information will be printed to the console.
quickload	Only valid if id="id", the annotation data will be fetched much quicker from the database.

Details

The annotation database table can be easily created in a PostgreSQL database using the dbUpdateAnnotation function.

Author(s)

Johannes Rainer

See Also

dbUpdateAnnotation dbGetExperimentInfo loadFromDB

`getEDB` *Retrieving expression values from a database.*

Description

`getEDB` retrieves expression values from the specified signal channels from a maDB database. For further informations and examples refer to the package vignette (which can be opened using the `openMadbVignette`).

Usage

```
getEDB(con,ids,signal.channels.pk,column.names=c("array_name","color"),v=TRUE)
```

Arguments

`con`	The connection to the database (created with the `dbConnect` function from the **RdbiPgSQL** package).
`ids`	The IDs of the genes from which the expression values should be retrieved. In the case of Affymetrix GeneChips the Affymetrix IDs should be used (as they are used as rownames in the `exprs` matrix in the `ExpressionSet` objects).
`signal.channels.pk`	the primary keys of the signal channels in the database from which the expression values should be retrieved. To get information about primary keys, arrays... from the database use the `dbGetExperimentInfo` function.
`column.names`	The information that should be used as column names of the expression values matrix. Allowed are all column names of the sample database table.
`v`	If TRUE additional information will be printed to the console.

Details

This function is useful, to retrieve only a subset of genes from a subset of signal channels of a microarray experiment (dataset) from a maDB database.

Author(s)

Johannes Rainer

See Also

publishToDB dbGetExperimentInfo loadFromDB getMDB

`dbGetExperimentInfo` *List information about an experiment from the database.*

Description

`dbGetExperimentInfo` lists all experiments that are stored in the maDB database, or returns informations from a particular experiment. Refer to the package vignette for furhter informations or examples (vignette can be opened using the `openMadbVignette` function).

Usage

```
dbGetExperimentInfo(Con,exp.title)
```

Arguments

`Con`	The connection to the maDB database (created using the `dbConnect` function from the **RdbiPgSQL** function).
`exp.title`	The experiment title (name) from the experiment which properties should be listet. If nothing is submitted a list with all experiments that are stored in the database is displayed.

Details

This function collects information about the specified experiment (database) by joining several database tables. A table is returned, that contains all primary keys or foreign keys with wich all data can be accessed for the particular experiment from the various tables. This tables are:

experiments Primary information about the experiment

arrays the arrays od GeneChips which where used in the experiment

signal_channels in two color micro arrays two signal channels exist and therefore two entries in the signal_channels table reference to one entry in the arrays table. For Affymetrix GeneChips the reference is one to one.

exp_values this table contains all the expression values from the corresponding signal channels of the experiment. The reference to the signal_channels table is established by the signal_channels_fk column in the exp_values table.

samples Each entry in this table references to one or more entries in the signal_channels table. This table contains informations about the samples that were hybridized onto the arrays.

Author(s)

Johannes Rainer

See Also

`dbGetAnnotation` `publishToDB` `dbGetComparisons` `MadbSet-class`

getMDB *Retriev regulation values from a database.*

Description

getMDB this function reads the regulation (M) values from the selected comparisons out of a maDB database. For further informations and examples refer to the package vignette (which can be opened using the openMadbVignette).

Usage

```
getMDB(con,ids=NULL,comparisons.pk=NULL,v=TRUE,column.names="description")
```

Arguments

con	The connection to the database (returned by the dbConnect function of the **RdbiPgSQL** package).
ids	The IDs of the genes from which the expression values should be retrieved. In the case of Affymetrix GeneChips the Affymetrix IDs should be used (as they are used as rownames in the exprs matrix in the exprSet objects).
comparisons.pk	the primary keys of the comparisons in the database for which the regulation values should be exported. A list of available comparisons and informations which sampes are compared in the various comparisons the dbGetComparisons function can be used.
column.names	The information that should be used as column names of the returned regulation values matrix. Allowed are all column names of the sample and comparisons database tables.
v	If TRUE additional information will be printed to the console.

Author(s)

Johannes Rainer

See Also

publishToDB dbGetExperimentInfo dbGetComparisons loadFromDB getEDB

getPK *Getting the primary key of the appropriate object from the database*

Description

Returns the primary key of an object from the database. If in the corresponding database table no entry with the same attribute values (of the submitted object) exists, the values will be inserted into the database and the primary key of the new entry will be returned.

Usage

```
getPK(object, Con, table.name = NULL, all.fields = FALSE, ...)
```

Arguments

object	The data object.
Con	The connection to the database.
table.name	The database table name where objects from the type of the submitted objects are stored into.
all.fields	If all attributes have to match between the database entry and the object submitted.
...	some additional things.

Value

The primary key of the database entry.

Author(s)

Johannes Rainer

---

getPathFromFile *Get the path to a file*

Description

`getPathFromFile` returns the path to the file.

Usage

```
getPathFromFile(file)
```

Arguments

file	The file name (should include also the path to the file). As file delimiter (also in Windows!) the Unix file delimiter "/" should be used!

Author(s)

Johannes Rainer

See Also

createFile

Examples

```
getPathFromFile("/home/jo/tmp/somefile.txt")
```

getSamplesFromDB *Lists all or only specified samples from the database.*

Description

getSamplesFromDB This function lists all samples that are stored into a database with the publishToDB function, or lists all samples that match the pattern submitted.

Usage

```
getSamplesFromDB(Con,table.name="samples",pattern)
```

Arguments

Con	The connection to the database (returned by the dbConnect function of the **RdbiPgSQL** package).
table.name	The name of the table that contains the information about the samples.
pattern	A pattern that can be used to search for in all columns of the table.

Details

This function returns a table (data.frame/matrix) with all samples stored in the database, or searches for those entries, that correspond to the submitted pattern.

Author(s)

Johannes Rainer

See Also

publishToDB MadbSet-class

isDecimal *Is the number a cardinal or a decimal number?*

Description

isDecimal This function determines if a number is decimal, or if it has no decimal places.

Usage

```
isDecimal(x)
```

Arguments

x	the number or a list of numbers.

Details

Sometimes it is useful to know if a number has decimal places, or if it is cardinal. This funciton can be used on single numbers or on a vector of numbers to solfe this problem.

Author(s)

Johannes Rainer

Examples

```
isDecimal(2)
isDecimal(2.1)
isDecimal(c(1,2.3,2,3.4))
```

---

loadFromDB-methods *Load a dataset from the database*

Description

loadFromDB in package **maDBi** This method allows to load data sets, stored previously into the database using the publishToDB method, back into the R workspace. A list of available data sets in a database created by the maDB package is returned by the dbGetExperimentInfo function (by submitting the connection to the database as parameter, a connection to a PostgreSQL database can be established by the dbConnect function from the **RdbiPgSQL**). \ For further informations and examples refer to the package vignette (which can be opened using the openMadbVignette).

Usage

```
loadFromDB(object=NULL,connection=NULL,name=NULL,pk=NULL,v=TRUE,...)
```

Arguments

object	An empty MadbSet or Sample object, generated either by link{newMadbSet} or new("MadbSet") for MadbSet objects, or by new("Sample") for Sample objects.
connection	The connection object to the database.
name	Only used if a MadbSet should be loaded from the database. Name should be equal to the experiment name that was stored into the database (use dbGetExperimentInfo to get a list of all available experiments in the database).
pk	For MadbSet: if a subset of arrays should be loaded from one experiment. As pk the primary keys of the corresponding signal channels should be submitted (to get a list of all signal channels, arrays, samples of a specific experiment in the maDB database use the dbGetExperimentInfo function by submitting also the name of the experiment along with the connection to the database as a parameter).
v	If TRUE additional informations will be printed to the console.
...	Additional parameters. Not used yet.

Methods

object = "MadbSet" Loads a MadbSet object from the database.

object = "Sample" Loads a Sample object from the database, (submit new("Sample"))

Warning

The usage of this method for EexprSet objects is deprecated! Use the MadbSet objects instead!

maDB-package *Package to store and retrieve microarray experiments into/from a database.*

Description

The maDB package allows to store microarray experiments (preprocessed expresion values of one-color (Affymetrix) or two-color microarrays along with sample information and annotation) into a database. As database backend a PostgreSQL database is used, where all needed tables are created automatically once the first experiment is stored into a newly created database using the publishToDB method. The database itself consists of a variety of tables, where the microarray data is stored in a relational way, thus allowing to store two-color microarray data and data from one-color (Affymetrix) microarray experiments in the same database. \ Use openMadbVignette to open the documentation (vignette) of the package.

Details

Package:	maDB
Type:	Package
Version:	1.9.4
Date:	2007-05-23
License:	LGPL

The main function to create the database tables and to insert an experiment into the database is publishToDB. Preprocessed expresion values and sample informations of the submitted MadbSet object are stored into the database. The function loadFromDB can be used to fetch a whole microarray experiment, or a subset of an experiment from the databse. Other functions to access just jubsets of expression values are getEDB. \ The web interface madbWeb (written in PHP) to a microarray database generated with the maDB package is accessible at http://madb.i-med.ac.at/madbWeb. \ Additionally the maDB package provides also functions to create graphical visualizations of microarray data like the drawMA function to generate MA (differential expression against average expression) plots, or drawVolcanoPlot to generate volcano plots. In both plots the local point density can be color coded.

Author(s)

Johannes Rainer Maintainer: johannes.rainer@tcri.at

References

http://madb.i-med.ac.at/madbWeb

multiapply *Apply a function to two arrays*

Description

multiapply allows to apply functions row wise or column wise to two arrays.

Usage

```
multiapply(a,b,MARGIN,FUN,...)
```

Arguments

a	An array wich rows or columns should be submitted to the function.
b	An array wich rows or columns should be submitted to the function, the argument a will be passed as first argument to the function while the argument b will be passed as the second one.
MARGIN	Like the MARGIN argument in the `apply` function, 1 means submit the rows to the function, 2 the columns.
FUN	The function that should be applied. Any function that needs two input parameters.
...	Additional parameters for the function FUN.

Details

Like the `apply` function this function can be used to apply functions to the rows or columns of a matrix, without having to write `for` loops. If the result of the function FUN is numeric or a character the function returns an array containing the results, otherwise a list with the output of the function FUN.

Author(s)

Johannes Rainer

See Also

`apply`

Examples

```
## creating two arrays
a <- matrix(ncol=3,nrow=3)
a[1,] <- c(1,2,2)
a[2,] <- c(3,2,3)
a[3,] <- c(2,2,2)
b <- matrix(ncol=5,nrow=3)
b[1,] <- c(8,8,6,8,10)
b[2,] <- c(2,3,3,2,3)
b[3,] <- c(1,1,1,1,2)

## now we perform a wilcox unpaired test for each row.
multiapply(a=a,b=b,MARGIN=1,FUN=wilcox.test,paired=FALSE)
```

mean.center *Mean center a numerical vector*

Description

`mean.center` mean centers a numerical vector.

Usage

```
mean.center(x,log.scale=TRUE,sd.scale=FALSE,...)
```

Arguments

`x`	The numerical vector that should be mean centered.
`log.scale`	If the values are in log2 scale. (in this case the mean of the vector will be subtracted from each value, otherwise divided).
`sd.scale`	if TRUE the mean centered values will be divided with the standard deviation.
`...`	Additional parameters. Not yet used.

Details

This function can be used to mean center for example gene expression data.

Author(s)

Johannes Rainer

Examples

```
mean.center(c(1,2,3,2,1,3,5,3,5))
```

meanWithoutNA *Calculating the mean by excluding all NA and NULL values*

Description

`meanWithoutNA` This function calculates the mean from the data submitted by excluding all NA and NULL values.

Usage

```
meanWithoutNA(Data)
```

Arguments

`Data`	A vector containing the values from which the mean should be calculated. NA and NULL values are removed before the calculation.

Details

Simple function to calculate the mean from a values vector.

Author(s)

Johannes Rainer

Examples

```
data <- c(2,3,5,1,NA,NULL,3,NA)
data
meanWithoutNA(data)
```

---

medianWithoutNA *Calculating the median by excluding all NA and NULL values*

Description

`medianWithoutNA` This function calculates the median from the data submitted by excluding all NA and NULL values.

Usage

```
medianWithoutNA(Data)
```

Arguments

`Data`	A vector containing the values from which the median should be calculated. NA and NULL values are removed before the calculation.

Details

Simple function to calculate the median from a values vector.

Author(s)

Johannes Rainer

Examples

```
data <- c(2,3,5,1,NA,NULL,3,NA)
data
medianWithoutNA(data)
```

multiGrep *Perform a pattern search with a list of patterns*

Description

multiGrep: search for a list of patterns.

Usage

```
multiGrep(pattern,x,...)
```

Arguments

pattern	The string, or a vector of strings, that should be searched in the submitted data.
x	The data, any kind of string or vector.
...	Some additional parameters for the myGrep and grep commands that are executed inside the function.

Details

The functions performs a myGrep call for each of the patterns submitted and returns a vector with the indizes where the patterns where found inside the data vector.

Author(s)

Johannes Rainer

See Also

grep, myGrep

Examples

```
test <- c("where","is","the","pattern","in","this","vector","?")
multiGrep(pattern="patt",test)
test[multiGrep(pattern="patt",test)]

multiGrep(pattern=c("the","tor","is","where"),test)
test[multiGrep(pattern=c("the","tor","is","where"),test)]
```

myGrep *Perform a pattern search*

Description

myGrep This function is a wrapper function for the grep function from R, that converts the data into a character vector before calling grep on the data. The function returns the index or a list of indizes, where (at what position) the pattern can be found in the submitted data.

Usage

```
myGrep(pattern,x,exact.match=FALSE,...)
```

Arguments

pattern	The string that should be searched in the submitted data.
x	The data, any kind of string or vector.
exact.match	If the string to search has to be exactly the same as the pattern or not (if TRUE, it is equal to a simple ==).
...	Some additional parameters for the grep command.

Details

This function is not really differnt from the grep call from R (as it uses also the grep call ;-)).

Author(s)

Johannes Rainer

See Also

grep, multiGrep

newMadbSet *Create a new MadbSet object*

Description

Function to create a new MadbSet object. Returns either an empty MadbSet object, or, if an ExpressionSet, RGList or MAList is submitted, a MadbSet containing all data from the submitted object..

Usage

```
newMadbSet(x = NULL, ...)
```

Arguments

x	A optional ExpressionSet (**Biobase**), RGList (**limma**) or MAList (**limma**) object.
...	some additional parameters.

Details

This function facilitites the creation of MadbSet objects. It takes either an ExpressionSet, RGList or MAList as input and returns a MadbSet containig all data from the submitted object.

Value

Returns a MadbSet (eventually filled with all the data from the submitted x, if x is an ExpressionSet, RGList or MAList)

Note

For further informations and examples refer to the package vignette (which can be opened using the `openMadbVignette`).

Author(s)

Johannes Rainer

See Also

`MadbSet-class`

`openMadbVignette` *maDB vignette*

Description

Opens the vignette for the maDB package.

Usage

```
openMadbVignette()
```

Author(s)

Johannes Rainer

`plotSimilarity` *Draw a plot for the Similarity-class*

Description

This function draws a similarity plot for the `Similarity-class` objects. For further informations and examples refer to the package vignette (which can be opened using the `openMadbVignette`).

Usage

```
plotSimilarity(ids, distances, data, template, restrict.to = min(5, length(data[
```

Arguments

`ids`	A list with the ids (rownames of the data matrix).
`distances`	The vector with the distances (each row of the data matrix to the template vector).
`data`	The data matrix (the expression or regulation patterns of a set of genes over a set of samples).
`template`	The template vector (the expression or regulation pattern of a gene over a set of samples).
`restrict.to`	How many patterns should be drawn. By default the 5 patterns with the smalles distance to the template pattern will be drawn.
`template.id`	The template gene id.
`legend`	If a legend should be drawn.
`...`	additional parameters.

Value

A plot of best matching gene expression or regulation patterns.

Author(s)

Johannes Rainer

See Also

`Similarity-class`

`progress` *Progress bar*

Description

`progress` displays a simple progress bar.

Usage

```
progress(percent=0,steps=20,type="default",label=TRUE)
```

Arguments

`percent`	The progress of an action in percent.
`steps`	The number of steps of the progress bar. Setting the steps to 20 means that each 5 percent to progress bar increases.
`type`	The type of the progress bar. Allowed types are “default” and “arrow”.
`label`	if the actual percentage should be written as a number too.

Details

This function can be used to display the progress of functions.

Author(s)

Johannes Rainer

See Also

`arrowProgress`

Examples

```
progress(percent=70,steps=40)
```

`publishToDB` *Writing the expression values from normalized micro arrays into a database*

Description

`publishToDB` allows to store a microarray experiment (dataset) into a database. \ For further informations and examples refer to the package vignette (which can be opened using the `openMadbVignette`).

Usage

```
publishToDB(object=NULL,connection=NULL,exp.name=NULL,exp.date=NULL,samples=NULL
```

Arguments

object	The object from which the expression values (preprocessed intensity values) should be stored in the database (an object from the type MadbSet).
connection	The database connection, with the dbConnect function from the **RdbiPgSQL** package.
exp.name	The experiment name (project name or dataset name). This name has to be unique in the database!
exp.date	The date of the experiment. Optional; as additional information the publishing date will automoatically be inserted in the database.
samples	A list of, or a single representation of a Sample object, that contains information about the samples hybridized on the chips. If the sample already exists in the database, it will not be added again. This parameter can also be omitted (but then no sample information will be available for this array!)
signal.channels	The linking element between the signal channels (columns in the exprs slot of the MadbSet object), and the corresponding samples hybridized on the specific signal channel. A list of SignalChannels of a MadbSet is returned by calling the function getSignalChannels on the object. If a list of Samples are submitted, the linkage between the signal channels and the samples has to be genereated by setting the sample.index slot of the SignalChannels appropriately.
preprocessing	A string giving information about the preprocessing of the microarray experiment (e.g. RMA).
pubmed.id	Optional pubmed id of the microarray experiment (if it has already been published).
global.transaction	If a global transaction should be used for the whole funciton call. The benefit of this is, that if any errors occurr during the function call, no data will be written into the database. It is always a good idea to send a transaction rollback call to the database (e.g. dbSendQuery(Con,"ROLLBACK TRANSACTION;")) if any exceptions occurred, to be shure that the database will not get inconsitent.
v	If TRUE additional informations will be printed to the console.

Author(s)

Johannes Rainer

See Also

getSignalChannels MadbSet-class SignalChannel-class Sample-class getPK loadFromDB

`replaceAll` *Replace all occurences of a specified character in a string by a new character*

Description

`replaceAll` This function helps to replace characters in a string or vector of strings.

Usage

```
replaceAll(x,old,new)
```

Arguments

`x`	The string or vector of strings where the replacements should occurr.
`old`	The character or vector of characters that should be replaced.
`new`	The character or vector of characters with whom the old characters should be replaced.

Details

This function replaces single characters with others.

Author(s)

Johannes Rainer

See Also

`replaces`

Examples

```
test <- c("zes","i have","an american","kezboard lazout")
replaceAll(test,old="z","y")
replaceAll(test,old=c("z","h"),new=c("y","d"))
```

`replaces` *Replace all occurences of a specified character in a string by a new character*

Description

`replaces` This function helps to replace characters in a string.

Usage

```
replaces(x,old,new,...)
```

Arguments

x	The strings where the replacements should occurr.
old	The character or vector of characters that should be replaced.
new	The character or vector of characters with whom the old characters should be replaced.
...	Additional parameters. Not used yet.

Details

This function replaces character strings with the one submitted. This function can only be applied to single strings. Use the `replaceAll` to apply it to vectors of strings.

Author(s)

Johannes Rainer

See Also

`replaceAll`

Examples

```
replaces("there are to much white spaces",old=" ","+")
```

NormChips *example exprSet instance*

Description

This data set is a small excerp from an Affymetrix hgu133plus2 microarray experiment. It contains 8 chips from 4 patients, 2 B-cell-ALL (acute lymphoblastic leukemia) and 2 T-cell-ALL patients. Two samples per patient were hybridized onto the chips, one before treatment and one after 6 hours GC (glucocorticoid) treatment. From every Affymetrix GeneChip only 1000 probe sets are included in this small dataset. \ For further informations and examples refer to the package vignette (which can be opened using the `openMadbVignette`).

Format

An `ExpressionSet` containing the expression values from 1000 genes on 8 different Affymetrix GeneChips.

toEexprSet *Transfor objects to objects of the type EexprSet*

Description

This function should transform data objects like limmas RGList or MAList objects into an instance of the class "EexprSet".

Usage

```
toEexprSet(object)
```

Arguments

object	The data object to transform.

Value

An instance of the type "EexprSet" that holds the information from the object submitted.

warning

This method as well as the EexprSet are deprecated! Use MadbSet and newMadbSet instead!

Author(s)

Johannes Rainer

See Also

EexprSet-class, createEexprSetFromLimma

dbUpdateAnnotation *Create and update an annotation table in a PostgreSQL database.*

Description

dbUpdateAnnotation creates and updates the annotation database table in a maDB database. \ For further informations and examples refer to the package vignette (which can be opened using the openMadbVignette).

Usage

```
dbUpdateAnnotation(Con,data,chip,date,do.backup=TRUE,v=TRUE)
```

Arguments

Con	The connection to the maDB database (returned by the function dbConnect of the **RdbiPgSQL** package).
data	The annotation table. Supported column namesare "id","gen_bank","description","uni (also additional columns containing some additional annotations are allowed, any ordering of columns is allowed).
chip	An identifier for the chip or array (oligo set) respectiveley.
date	The date when the annotation was created.
do.backup	If a backup of the annotation table should be done. This is usefull when the annotation of an already existing table is going to be updated. By default the annotations of the chip that will be updated will be saved as a tab delimited txt file. The backup file will be saved in the current workspace.
v	If TRUE additional information will be printed to the console.

Details

When creating or updating the database not all columns have to be submitted in the data argument, when submitting a table with only some of the columns the other columns in the database will remain empty.

Author(s)

Johannes Rainer

See Also

dbGetAnnotation

---

writeFigure *Includes a figure in a LaTeX document*

Description

writeFigure This function writes all the needed lines into a tex document to include an image into a LaTeX document (the image has to be created before.).

Usage

```
writeFigure(file=NULL,image=NULL,label=NULL,caption=NULL,option="[!ht]",image.wi
```

Arguments

file	The file where the lines should be included. If the file does not exist it will be created, if the file should be in some subdirectories, these will also be created if needed. As file a file handle or a character string giving the file name (and or directories) can be submitted.
image	The path to the image (relative from the directory where the file lies into).
label	The label that should be used to link to this figure.
caption	The caption that should be written below the figure.
option	Some LaTeX specific options for the figure.
image.width	The image width relative to the text width.

Details

The lines that are needed to include an image into a LaTeX document are inserted with this function into a file, if the file does already exist, the lines will be appended. As file parameter also a file handle can be submitted.

Author(s)

Johannes Rainer

See Also

writeLatexTable

---

writeLatexTable *Writes a matrix or data frame in latex format to a file.*

Description

writeLatexTable This function writes a matrix or a data frame formatted as a LaTeX table to a text file. The text size as well as the number of digits and the desired caption and label can be specified.

Usage

```
writeLatexTable(data=NULL,file=NULL,append=TRUE,textsize="normalsize",caption=NU
```

Arguments

data	The data that should be written. A matrix or a data frame, the column names will be used as column headers, the row names as row headers.
file	The file where the table should be written into. If directories are specified, these directories (if not already existant) will be created. As file parameter also a file handle can be submitted.
append	If the table should be appended to the text in the file.
textsize	The size of the text, possible alternatives for "normalsize" are "small", "scriptsize" and all other LaTeX text sizes.
caption	The text that should be written into the table caption.
label	The label text that can be used to link to this table.
digits	The number of digits that should be used for all numeric values in the table.

Details

One important feature of this function is, that the table can be written into a file that has not to be in the working directory. So one can for example by submitting as file="doc/tabletest.tex" writing the table into the file "tabletest.tex", that is in the subdirectory "doc" in the working directory. If this directory does not already exist, the function will create it. But remember that you have to use always the unix file separator "/", also in Windows!

Author(s)

Johannes Rainer

See Also

`writeFigure`

Index