Package 'MAMA'

February 19, 2015

Version 2.2.1

Date 2013-1-28

Title Meta-Analysis of MicroArray

Author Ivana Ihnatova <184415@mail.muni.cz>.

Maintainer Ivana Ihnatova <184415@mail.muni.cz>

Depends R (>= 2.15.1), methods, genefilter, metaMA, xtable, multtest, gtools, grid, GeneMeta

Suggests gplots, RankProd

Imports MergeMaid, GeneMeta, xtable, methods, metaArray

Description Implementation of methods for microarray meta-analysis.

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2013-04-30 06:22:29

R topics documented:

`clinical` *Functions to retrieve and assign*

Description

Functions access the individual slots of an object derived from 'MetaArray' class.

Usage

```
clinical(object)
clinical(object)<-value

GEDM(object)
GEDM(object)<-value

datanames(object)
datanames(object)<-value
```

Arguments

`object`	An object derived from 'MetaArray' class
`value`	A list of gene expression data matrices, clinical data matrices or a vector of data names

Value

'clinical' returns the list of clinical data matrices, one for each data set; 'GEDM' returns the list of gene expression data matrices, one for each data set; 'datanames' returns the vector of data sets names

Author(s)

Ivana Ihnatova

`clinical.sum` *Function to calculate summaries of clinical data*

Description

Function calculates summaries of clinical data in object of class MetaArray. Absolute and relative frequencies of factors and descriptive statistic (minimum, median, mean, quartiles, maximum) are provided for continuous variables. Overall summaries for all datasets are also provided.

Usage

```
clinical.sum(x)
```

Arguments

`x`	An object of class MetaArray

Value

`absolute`	A list of absolute frequencies or desciptive statistics, one slot refers to one variable
`realative`	A list of relative frequencies, one slot refers to one variable

Author(s)

Ivana Ihnatova

Examples

```
data(ColonData)
clinical.sum(ColonData)
```

`colIntersect` *Function to find intersect in columns of a data.frame*

Description

Function returns intersect of all coulmns of a data.frame.

Usage

```
colIntersect(x)
```

Arguments

`x`	A data.frame

Details

Intersect is found recursively. In means that at first the intersect of the first and the second column is computed. Later this intersect is compared to third column in order to obtain common values etc.

Value

Vector of common values

Author(s)

Ivana Ihnatova

See Also

`intersect`, ~~~

Examples

```
genes<-paste("Gene", 1:100)
O<-cbind(sample(genes), sample(genes), sample(genes))
colIntersect(O[1:50,])
```

`ColonData` *Example dataset for meta-analysis of microarray*

Description

This is an example dataset for meta-analysis of microarray. It has been created from three datasets form Gene Expression Omnibus (GSE13067, GSE13294 and GSE4554). The data have been normalized, log2-transformed and only random selection of 500 gene is included.

Usage

```
data(ColonData)
```

Format

The format is: Formal class 'MetaArray' [package "MAMA"] with 3 slots ..@ GEDM :List of 3 .. ..$: num [1:500, 1:77] 2.49 3.87 2.95 6.39 6.06 ... .- attr(*, "dimnames")=List of 2 .. ..$: chr [1:500] "217562_at" "203766_s_at" "1554394_at" "212662_at" ... .$: chr [1:77] "GSM335574" "GSM335645" "GSM335546" "GSM335623" ... .$: num [1:500, 1:36] 3.24 5.32 3.24 7.24 4.75 ... .- attr(*, "dimnames")=List of 2 .. ..$: chr [1:500] "217562_at" "203766_s_at" "1554394_at" "212662_at" ... .$: chr [1:36] "GSM327331" "GSM327282" "GSM327313" "GSM327353" ... .$: num [1:500, 1:41] 0.595 3 1.618 4.668 2.887 ... .- attr(*, "dimnames")=List of 2 .. ..$: chr [1:500] "217562_at" "203766_s_at" "1554394_at" "212662_at" ... .$: chr [1:41] "GSM101849" "GSM101851" "GSM101857" "GSM101799" ... ..@ clinical :List of 3 .. ..$:'data.frame': 77 obs. of 1 variable: .. .. ..$ satelite: Factor w/ 2 levels "MSI","MSS": 1 1 1 1 1 1 1 1 1 1 ... .$:'data.frame': 36 obs. of 1 variable: .. .. ..$ satelite:

Factor w/ 2 levels "MSI","MSS": 1 1 1 1 1 2 2 2 2 2 ... .$:'data.frame': 41 obs. of 2 variables: .. .. ..$ position: Factor w/ 3 levels "distal","proximal",..: 2 1 2 2 1 3 2 2 1 2 ... .$ satelite: Factor w/ 2 levels "MSI","MSS": 1 1 1 1 1 1 1 1 1 1 ... ..@ datanames: chr [1:3] "denmark" "australia" "japan"

Source

http://www.ncbi.nlm.nih.gov/geo/

Examples

```
data(ColonData)
plot(ColonData)
```

`commonGenes` *Function to compute number of common genes in ordered gene lists*

Description

Function computes number of common genes up to each position (from `1` to `n`)

Usage

```
commonGenes(ord, n)
```

Arguments

`ord`	Data frame, where columns refer to ordered gene list from one study
`n`	The last position to be concered

Value

Numeric vector, number of common genes up to each position

Note

Created as part of implementation of the Similarity of Ordered Gene Lists method

Author(s)

Ivana Ihnatova

References

Yang, X., Bentink, S., Scheid, S. Spang, R., Similarities of ordered gene lists, 2005

Examples

```
genes<-paste("Gene", 1:100)
O<-cbind(sample(genes), sample(genes), sample(genes))
commonGenes(O,100)
```

`compute.RQ` *Function to compute R and Q statistics as defined in - see References*

Description

Function computes R (average rank across studies) and Q (sum of the squared deviations of each study's rank for the gene from the mean of the ranks for that gene)

Usage

```
compute.RQ(RAN)
```

Arguments

`RAN`	matrix with rank of genes as produced by `rank.genes`, with rows coresponding to genes and columns coresponding to studies

Value

matrix with first column of R statistic and second of Q statistic

Author(s)

Ivana Ihantova

References

Zintzaras, E., Ioannidis, J.P.A 2008 Meta-analysis for ranked discovery datasets: Theoretical framework and empirical demonstration for microarrays, Computational Biology and Chemistry 32, 39-47

See Also

`rank.genes`,`MCtest`

Examples

```
RANK<-cbind(sample(100), sample(100), sample(100))
RQ<-compute.RQ(RANK)
head(RQ)
```

computeAlpha *Function to do compute tunning parameter alpha*

Description

Function computes vector of possible alphas in Similarity of Ordered Gene List method. See Details.

Usage

```
computeAlpha(n = NULL, min.weight = 1e-05, ngenes)
```

Arguments

`n`	Number of genes to be considered in the comparison , if `NULL` a pre-defined vector is used
`min.weight`	Minimal weight to be counted
`ngenes`	Number of genes in the dataset

Details

Alphas are calculated so that at certain position (`n`), the exponential weights reach `min.weight`. If one is interessted in comparing ordered gene lists up to certain position, alpha appropriate for this position can be calculated.

Value

Numeric vector of possible alphas

Author(s)

Ivana Ihnatova

References

Yang, X., Bentink, S., Scheid, S. Spang, R., Similarities of ordered gene lists, 2005

Examples

```
#using default n
A<-computeAlpha(ngenes=1000)

#or with user-selected n
A<-computeAlpha(n=seq(from=25, to=300, by= 25),ngenes=1000)
```

`computeOrdering` *Function to compute ordered gene lists*

Description

Function computes test statistic for each gene in each dataset of MetaArray object and orders them form the most up-regulated (possitive statisics) to the most down-regulated (negative statistics).

Usage

```
computeOrdering(data, varname, test)
```

Arguments

`data`	MetaArray object
`varname`	A string indicating which column of clinical data matrices should be used to compute test statistic. Same column is used in all datasets.
`test`	`"FCH"` for fold change (function `fold.change`) or `"T"` for T-test (function `meta.test`)

Value

A data frame, each column refers to ordered gene list from one study

Author(s)

Ivana Ihnatova

See Also

`fold.change`, `meta.test`

Examples

```
data(Singhdata)

cl1<-as.data.frame(Singhdata$classes[[1]])
names(cl1)<-"classlab"
cl2<-as.data.frame(Singhdata$classes[[2]])
names(cl2)<-"classlab"
cl3<-as.data.frame(Singhdata$classes[[3]])
names(cl3)<-"classlab"
rownames(Singhdata$esets[[1]])<-Singhdata$geneNames
rownames(Singhdata$esets[[2]])<-Singhdata$geneNames
rownames(Singhdata$esets[[3]])<-Singhdata$geneNames

data<-new("MetaArray", GEDM=list(Singhdata$esets[[1]], Singhdata$esets[[2]], Singhdata$esets[[3]]),
clinical=list(cl1, cl2, cl3), datanames=c("dataset1", "dataset2", "dataset3"))
```

```
ord<-computeOrdering(data, "classlab", "FCH")
```

`conting.tab` *Contingency table from gene lists*

Description

Function to make a contingency table from gene lists as in VennMapper program.

Usage

```
conting.tab(lists)
```

Arguments

`lists`	list of vectors. Each vector refers to a method and contains names of significant genes

Details

Simmilar to `gene.list` and `Z`, but provides different output

Value

Matrix with counts of matches in pairs of gene lists

Author(s)

Ivana Ihnatova

References

Smid, M., Dorssers, L. C. J. and Jenster, G. 2003, Venn Mapping: clustering of heterologous microarray data based on the number of co-occurring differentially expressed genes, Bioinformatics, vol. 19 no. 16 2003

See Also

`Z`,`gene.list`

Examples

```
lists<-list(Method1=c("Gene_A", "Gene_V","Gene_S","Gene_C","Gene_U","Gene_D","Gene_E","Gene_G","Gene_W"),
  Method2=c("Gene_D","Gene_W","Gene_G","Gene_E","Gene_H","Gene_X"),
  Method3=c("Gene_L","Gene_K","Gene_J","Gene_M","Gene_V","Gene_T","Gene_R","Gene_U"))
conting.tab(lists)
```

cv.filter *Microarray probes filtering*

Description

Function to filter microarray probes according to coefficient of variation

Usage

```
cv.filter(data, cutoff = 0.05)
```

Arguments

`data`	expression matrix with probes in rows and samples in columns
`cutoff`	cutoff value for filtering

Value

Expression matrix, probes with CV below cutoff are filtered out.

Author(s)

Ivana Ihnatova

Examples

```
data(Singhdata)
data<-Singhdata$esets[[1]][1:1000,]
data.filtered<-cv.filter(data)
head(data.filtered)
```

entitybuild2 *Function to calculate test statistic for microarray data*

Description

Calculates test statistic for microarray data

Usage

```
entitybuild2(expr.mat, ALLtype = NULL, type, dataset = NULL, minSampleNum = 3, method = "t", random = 
```

Arguments

`expr.mat`	Expression matrix, with rows corresponding to genes and columns to samples
`ALLtype`	Vector of class labels, must be `factor`
`type`	Levels of class labels
`dataset`	Name of the dataset
`minSampleNum`	Minimal number of samples required for test statistic
`method`	Type of test as in `mt.teststat` (one of `fc`, `t`, `z`)
`random`	Logical, if TRUE samples are assinged to groups randomly

Value

Vector of test statistics.

Author(s)

Code provided by Xinan Yang <xnyang@seu.edu.cn> has been modified by Ivana Ihnatova

References

Yang, X., Bentink, S. a Spang, R. 2005, Detecting Common Gene Expression Patterns in Multiple Cancer Outcome Entities, Biomedical Microdevices, vol.7:3

Examples

```
data(Singhdata)
group<-as.factor(Singhdata$classes[[1]])
entitybuild2(Singhdata$esets[[1]], ALLtype=group, type=levels(group))
```

`ES.GeneMeta` *Wrapper function for combining the effect size as implemented in GeneMeta package*

Description

This is a wrapper function for meta-analytical method implemented in GeneMeta package

Usage

```
ES.GeneMeta(data, varname, useREM = TRUE, CombineExp = 1:length(esets), nperm = 1000)
```

Arguments

`data`	MetaArray object
`varname`	Character String - name of one column in clinical data matrices to be used as class labels
`useREM`	Logical - indicating whethet Random Effect Model (REM) shuld be used, if `FALSE` then Fixed Effect Model is applied
`CombineExp`	A numeric vector - which experiments should be combined, all experiments are set as default
`nperm`	Number of permutations to calculate FDR

Value

An object of class `ES.GeneMeta.res`

`theScores`	Ouput from function `zScores`
`ScoresFDR`	Output from function `zScoreFDR`

Author(s)

Ivana Ihnatova

References

Choi et al, Combining multiple microarray studies and modeling interstudy variation. Bioinformatics, 2003, i84-i90.

See Also

`zScores`, `zScoreFDR`

Examples

```
data(ColonData)
es<- ES.GeneMeta(ColonData, "MSI", nperm = 10)
```

`flip` *Function to flip data frames*

Description

Function reverses the order of rows. It is simmilar to function `rev`, but designed for rows of a data frame, matrix.

Usage

```
flip(order)
```

Arguments

`order`	Data frame, Matrix

Value

Same data frame or matrix with reversed rows

Author(s)

Ivana Ihnatova

See Also

`rev`

Examples

```
A<-matrix(1:24, ncol=4);A
flip(A)
```

`fold.change` *Function to do compute fold change between two groups*

Description

Function computes fold change between two groups of log2-transformed data

Usage

```
fold.change(x, varname)
```

Arguments

`x`	MetaArray object
`varname`	Character String specifying which column of clinical data matrices should be used as class labels. Column of this name must be present in each clinical data matrix.

Value

Data frame of fold changes, each column refer to one study and row to genes.

Author(s)

Ivana Ihnatova

Examples

```
#data preparation
data(Singhdata)
cl1<-as.data.frame(Singhdata$classes[[1]])
names(cl1)<-"classlab"
cl2<-as.data.frame(Singhdata$classes[[2]])
names(cl2)<-"classlab"
cl3<-as.data.frame(Singhdata$classes[[3]])
names(cl3)<-"classlab"

rownames(Singhdata$esets[[1]])<-Singhdata$geneNames
rownames(Singhdata$esets[[2]])<-Singhdata$geneNames
rownames(Singhdata$esets[[3]])<-Singhdata$geneNames

dataset<-new("MetaArray", GEDM=list(Singhdata$esets[[1]], Singhdata$esets[[2]], Singhdata$esets[[3]]),
clinical=list(cl1, cl2, cl3), datanames=c("dataset1", "dataset2", "dataset3"))

#fold change
fch<-fold.change(dataset, "classlab")
head(fch)
```

`GEDM<--methods` *Replacement Methods for* `MetaArray` *object*

Description

Each of the methods replaces one slot of an object derived from `MetaArray` class

Methods

`signature(object = "MetaArray")` Method replaces one slot of an object derived from `MetaArray` class, e.g. `"GEDM<-"` replaces the `GEDM` slot etc.

`gene.list` *Intersect of gene lists*

Description

This function takes list of gene list as input and returns a matrix of gene names common in pairs of lists

Usage

```
gene.list(lists)
```

Arguments

`lists`	list of vectors. Each vector refers to a method and contains names of significant genes

Details

Simmilar to `conting.tab` and `Z`, but provides different output

Value

A matrix of gene names common in two gene lists

Author(s)

Ivana Ihnatova

See Also

`conting.tab`, `Z`

Examples

```
lists<-list(Method1=c("Gene_A", "Gene_V","Gene_S","Gene_C","Gene_U","Gene_D","Gene_E","Gene_G","Gene_W"),
  Method2=c("Gene_D","Gene_W","Gene_G","Gene_E","Gene_H","Gene_X"),
  Method3=c("Gene_L","Gene_K","Gene_J","Gene_M","Gene_V","Gene_T","Gene_R","Gene_U"))
gene.list(lists)
```

`gene.select.FC` *Function to select genes according to fold change*

Description

Function selects genes with fold change (in absolute value) above input cutoff

Usage

```
gene.select.FC(fch, cutoff)
```

Arguments

`fch`	Data frame of fold change with columns corresponding to microarray experiments and rows to genes
`cutoff`	Cutoff for selection

Value

List - each slot refers to one column of input data frame and it is a vector of genes names with fold change above selected threshold

Author(s)

Ivana Ihantova

Examples

```
#data preparation
data(Singhdata)
cl1<-as.data.frame(Singhdata$classes[[1]])
names(cl1)<-"classlab"
cl2<-as.data.frame(Singhdata$classes[[2]])
names(cl2)<-"classlab"
cl3<-as.data.frame(Singhdata$classes[[3]])
names(cl3)<-"classlab"

rownames(Singhdata$esets[[1]])<-Singhdata$geneNames
rownames(Singhdata$esets[[2]])<-Singhdata$geneNames
rownames(Singhdata$esets[[3]])<-Singhdata$geneNames

dataset<-new("MetaArray", GEDM=list(Singhdata$esets[[1]], Singhdata$esets[[2]], Singhdata$esets[[3]]),
clinical=list(cl1, cl2, cl3), datanames=c("dataset1", "dataset2", "dataset3"))

#fold change
fch<-fold.change(dataset, "classlab")
#gene selection
genes.selected<-gene.select.FC(fch, 1)
```

---

join.DEG *Function to join vectors of differentially expressed genes to one list*

Description

The function takes outputs from meta-analysis of microarrays, extracts names of differentially expressed genes from them and joins these names into one list, where each slot refer to one output.

Usage

```
join.DEG(..., genenames, type = NULL, cutoff)
```

Arguments

`...`	Outputs from different function for methods of meta-analysis of microarray
`genenames`	a character vector - names of all genes (or probe ID) included in meta-analysis. It can be `NULL` if the wrapper functions were used for the analysis.
`type`	a numeric vector idicating from which function the output is, kth element in `type` corresponds to kth element of `...`. It is not needed when wrapper functions where used.
`cutoff`	a numeric value - a cutoff level for p-value to select significant genes

Details

Values below have to be used in `type`.

- 1for functions: `pvalcombination`, `pvalcombination.paired`, `EScombination` or `EScombination.paired`
- 2for function `zScores`
- 3for function `ScoresFDR`
- 4for function `performSOGL`
- 5for function `topGene`
- 6for function `z.stat`
- 7for function `MAP.genes`

Value

A list in which each slot refers to one meta-analytical method and contains names of differentially expressed genes found by the method.

Author(s)

Ivana Ihnatova

`join.results` *Function to join results from meta-analysis to one list*

Description

Function joins results from meta-analysis to one list. It uses predefined types of results and transform some of them.

Usage

```
join.results(..., type = NULL , genenames = NULL)
```

Arguments

`...`	Outputs from different function for methods of meta-analysis of microarray
`type`	a numeric vector idicating from which function the output is, kth element in `type` corresponds to kth element of `...`. It can be `NULL`, if the wrapper functions were used.
`genenames`	a character vector - names of all genes (or probe ID) included in meta-analysis = rownames of gene expression data matrix. It can be `NULL`, if the wrapper functions were used.

Details

Values below have to be used in `type`.

- 1for functions: `pvalcombination`, `pvalcombination.paired`, `EScombination` or `EScombination.paired`
- 2for function `performSOGL`
- 3for function `topGene`
- 4for function `MAP.genes`
- 5for function `zScores`, `ScoresFDR`, `z.stat`, `tspcalc`

Value

A list in which each slot refers to one meta-analytical method and contains a data frame with all outputs available from the method for one gene.

Author(s)

Ivana Ihnatova

`make.matrix` *Function to make matrix for heatmap to compare results of several methods*

Description

`make.matrix` returns matrix of 1 and 0 with gene names as rows and methods as colums. 1 refers to the gene that was found as differentialy expressed by the method, othterwise 0 is used.

Usage

```
make.matrix(lists)
```

Arguments

`lists`	list of vectors. Each vector refers to a method and contains names of significant genes

Value

Binary matrix with gene names as rows and methods as colums.

Author(s)

Ivana Ihnatova

Examples

```
lists<-list(Method1=c("Gene_A", "Gene_V","Gene_S","Gene_C","Gene_U","Gene_D","Gene_E","Gene_G","Gene_W"),
  Method2=c("Gene_D","Gene_W","Gene_G","Gene_E","Gene_H","Gene_X"),
  Method3=c("Gene_L","Gene_K","Gene_J","Gene_M","Gene_V","Gene_T","Gene_R","Gene_U"))
make.matrix(lists)
```

`MAP.genes` *Function to do assign probesets IDs to patterns*

Description

Function makes a list of vectors of probeset IDs. One vector contains probesets with one observed pattern.

Usage

```
MAP.genes(resx, value.dis, files = TRUE)
```

Arguments

`resx`	data.frame, rows refer to patterns, columns to pattern description - see in examples
`value.dis`	Matrix of observed patterns: binary matrix, columns refer to studies, rows to genes,
`files`	logical, when TRUE, files with probeset IDs are written too

Value

list, each slot is vector of gene names

Author(s)

Ivana Ihnatova

Examples

```
#> t(resx)
#                        111 101    110    011
#n.sig[which(n.sig > 1)]   3   2  2.000  2.000
#n.strong                 32 127 20.000  6.000
#n.soft                   32 159 52.000 38.000
#p.soft                    0   0  0.000  0.000
#p.strong                  0   0  0.000  0.002
#permu.soft                0   0  0.000  0.000
#permu.strong              0   0  0.001  0.008
```

MAP.Matches *Wrapper function for MAP-Matches method*

Description

This is a wrapper function for MAP-Matches method.

Usage

```
MAP.Matches(data, varname, t.cutoff = "98.00%", multiple = TRUE, perm = c("both", "columns", "labels")
```

Arguments

`data`	Object of class `MetaArray`
`varname`	Character String - name of one column in clinical data matrices to be used as class labels
`t.cutoff`	Character String - quantile of T statistics to be selected, e.g. "95.00%" selects the top 5 percent of absolute values
`multiple`	Logical - when `TRUE` only paterrns with multiple '1' are used
`perm`	Character String - if `"labels"` only class labels are permuted for statistical analysis (empirical significance), if `"columns"` only genes in each dataset are selected randomly, if `"both"` both class labels and genes are permuted and two p-values returned
`nperm`	Numeric - number of permutations
`test`	Character String - if `"t"` then unequal variance t-test is used, if `"t.equalvar"` equal variance t-test is used
`sig.col`	Character String - which p-value is used for selection of significant patterns. Possible values are: `"p.col.strong"`, `"p.col.weak"`, `"p.lab.strong"`, `"p.lab.weak"` , "col" refers to column permutations, "lab" to class labels, "weak" to soft match and "strong" to strong match
`sig.cutoff`	Numeric - p-value for selection of sigificant patterns

Value

Object of class `MAP.Matches.res` containing

`tests`	Data.frame of test statistics
`bin.matrix`	Binary matrix from `tests`, `1` means the test statistics was higer than threshold
`sumarization`	Sumarization of `bin.matrix`: number of selected genes in each dataset, genes with at least one `1` in pattern, probability of observing strong or soft match in the data
`MAP`	Data frame describing observed patterns: number of strong `n.strong` and soft `n.soft` matches and number of genes involved `n.sig`
`stat.analysis`	Results of statistical analysis
`genes`	List of genes observed with each pattern
`all.genes`	Names of the all genes in the analysis

Author(s)

Ivana Ihnatova

References

Yang, X., Bentink, S. and Spang, R. 2005, Detecting Common Gene Expression Patterns in Multiple Cancer Outcome Entities, Biomedical Microdevices, Vol.7:3, pp. 247-251

Examples

```
data(ColonData)
MAP.Matches(ColonData, "MSI", nperm = 100, sig.col="p.lab.strong")
```

MAPmatrix *Function to summarize binary matrix*

Description

Function `MAPmatrix` summarizes a binary matrix. It treats each row as Meta-Analysis Pattern and looks for count of observed soft and strong matches.

Usage

```
MAPmatrix(value.dis)
```

Arguments

`value.dis`	A binary matrix with rows refering to genes and columns to microarray studies.

Value

A matrix with rows corresponding to MAP patterns and four columns: unique patterns that are being observed in the data (`uniqe.pat`), number of observed soft matches with the pattern (`n.soft`), number of observed strong matches (`n.strong`) and number of $1$'s in the pattern `n.sig`)

Author(s)

Ivana Ihnatova

References

Yang, X., Bentink, S. a Spang, R., *Detecting Common Gene Expression Patterns in Multiple Cancer Outcome Entities*, Biomedical Microdevices, Vol.7:3, 2005

Examples

```
## The function is currently defined as
function(value.dis)
{
res<-ratio(value.dis)
unique.pat <- unique(results$X.string)
n.soft <- patternMatch(value.dis,unique.pat)
n.strong <- patternMatch.strong(value.dis,unique.pat)
unique.X <- patternmatrix(unique.pat,ncol(value.dis))
n.sig <- apply(unique.X,1,sum)
mat<-data.frame(unique.pat, n.soft, n.strong,n.sig)
return(mat)
  }
```

`MAPsig1` *Pattern signifficance*

Description

Function computes significance of observed number of strong and soft matches by randomly choosing differentially expressed genes in each study.

Usage

```
MAPsig1(unique.pat, value.dis, iter = 1000)
```

Arguments

`unique.pat`	unique meta-analysis patterns
`value.dis`	binary matrix from T-statistics
`iter`	number of iteration

Value

data.frame with p-values for number of oberved soft and strong matches

Author(s)

Ivana Ihnatova

References

Yang, X., Bentink, S. and Spang, R. 2005, Detecting Common Gene Expression Patterns in Multiple Cancer Outcome Entities, Biomedical Microdevices, Vol.7:3, pp. 247-251

`MAPsig2` *Pattern significance*

Description

Function computes significance of observed number of strong and soft matches by randomly assigning group labels in each study.

Usage

```
MAPsig2(out,value.dis, unique.pat, B = 1000)
```

Arguments

`out`	output from function `test.group.shuffle`
`value.dis`	binary matrix from T-statistics
`unique.pat`	unique meta-analysis patterns
`B`	number of iterations

Value

data.frame with p-values for number of oberved soft and strong matches

Author(s)

Ivana Ihnatova

References

Yang, X., Bentink, S. and Spang, R. 2005, Detecting Common Gene Expression Patterns in Multiple Cancer Outcome Entities, Biomedical Microdevices, Vol.7:3, pp. 247-251

`MCtest` *Monte Carlo permutation test*

Description

This function performs Monte Carlo permutation test to asses the statistical significance of R and Q statistics.

Usage

```
MCtest(RAN, RQ, nper = 100)
```

Arguments

`RAN`	matrix of ranks to be permuted, columns refer to studies, rows refer to genes
`RQ`	observed values of R (average rank) and Q (heterogenity) - as produced by `compute.RQ`
`nper`	number of permutations

Value

Returns a matrix with four columns. First (Second) column represents significance level of high (low) avarage rank. Third (fourth) represents significance level of high (low) heterogenity.

Author(s)

Ivana Ihnatova

References

Zintzaras, E., Ioannidis, J.P.A 2008 Meta-analysis for ranked discovery datasets: Theoretical framework and empirical demonstration for microarrays, Computational Biology and Chemistry 32, 39-47

See Also

`rank.genes`,`compute.RQ`

Examples

```
RANK<-cbind(sample(100), sample(100), sample(100))
RQ<-compute.RQ(RANK)
head(RQ)
MCtest(RANK, RQ, nper=100)
```

`mergedata` *Function to merge data from MetaArray object*

Description

Function merges the data stored in MetaArray object. It binds expression data matrices into one gene expression data matrix. It creates one binary vector of class labels of the samples and one numeric vector of orgin of the samples.

Usage

```
mergedata(x, varname)
```

Arguments

`x`	MetaArray objec
`varname`	character String specifying the column of clinical data to be used in vector of class labels of the samples

Value

A list with three slots

`dat`	Gene expression data matrix, rows refer to genes/probes and columns to samples
`cl`	Binary vector of class labels of the samples
`origin`	Numeric vector describing the origin of the samples. Same number refers to samples from one study

Author(s)

Ivana Ihnatova

Examples

```
data(Singhdata)

cl1<-as.data.frame(Singhdata$classes[[1]])
names(cl1)<-"classlab"
cl2<-as.data.frame(Singhdata$classes[[2]])
names(cl2)<-"classlab"
cl3<-as.data.frame(Singhdata$classes[[3]])
names(cl3)<-"classlab"
rownames(Singhdata$esets[[1]])<-Singhdata$geneNames
rownames(Singhdata$esets[[2]])<-Singhdata$geneNames
rownames(Singhdata$esets[[3]])<-Singhdata$geneNames

data<-new("MetaArray", GEDM=list(Singhdata$esets[[1]], Singhdata$esets[[2]], Singhdata$esets[[3]]),
clinical=list(cl1, cl2, cl3), datanames=c("dataset1", "dataset2", "dataset3"))

merged.data<-mergedata(data,"classlab")
summary(merged.data)
```

`mergeExprs2` *Function to merge ExpressionSet object*

Description

Function `mergeExprs` from library `MergeMaid` has been modiffied to have to arguments: list of ExpressionSet objects and vector of datasets names.

Usage

```
mergeExprs2(arg, names)
```

Arguments

`arg`	List of ExpressionSet objects
`names`	Vector of datasets names

Value

A mergeExpressionSet object.

Author(s)

Ivana Ihnatova

`meta.test` *Function to compute T-statistic and p-value in meta-analysis*

Description

Function `meta.test` returns a list with two slots: data frame of test statistics and data frame of p-values. In each of the matrices rows correspond to genes and columns to data sets.

Usage

```
meta.test(x, varname, stat = "t")
```

Arguments

`x`	MetaArray object
`varname`	A String indicating which column of clinical data matrices should be used as class labels. Column of such name must be present in all datasets. It must not be a binary vector (0's and 1's)
`stat`	A character String indicating the type of test statistic to be computed as used in `mt.teststat` function

Value

A list with two slots:

`test`	A data frame of statistics in which rows correspond to genes and columns to data sets
`p`	A data frame of p-values (only if `test="t"` returned) in which rows correspond to genes and columns to data sets

Author(s)

Ivana Ihnatova

Examples

```
data(Singhdata)

cl1<-as.data.frame(Singhdata$classes[[1]]+1)
names(cl1)<-"classlab"
cl2<-as.data.frame(Singhdata$classes[[2]]+1)
names(cl2)<-"classlab"
cl3<-as.data.frame(Singhdata$classes[[3]]+1)
names(cl3)<-"classlab"
rownames(Singhdata$esets[[1]])<-Singhdata$geneNames
rownames(Singhdata$esets[[2]])<-Singhdata$geneNames
rownames(Singhdata$esets[[3]])<-Singhdata$geneNames

data<-new("MetaArray", GEDM=list(Singhdata$esets[[1]], Singhdata$esets[[2]], Singhdata$esets[[3]]),
clinical=list(cl1, cl2, cl3), datanames=c("dataset1", "dataset2", "dataset3"))

m<-meta.test(data,"classlab")
```

`MetaArray-class` *Class "MetaArray" ~~~*

Description

A class created for meta-analysis of microarray

Objects from the Class

Objects can be created by calls of the form `new("MetaArray", ...)`.

Slots

`GEDM`**:** Object of class `"list"` - gene expression data matrices are stored in individual slot of the list. Each matrix refer to one dataset, genes are represented in rows, samples in columns.

`clinical`**:** Object of class `"list"` - clinical data matrices, clinical description of samples, rows refer to samples, columns to clinical characteristics

`datanames`**:** Object of class `"character"` - vector of names of the datasets

Methods

plot `signature(x = "MetaArray", y = "missing")`: draws distribution of clinical variables of several datasets. Boxplot is drawn for numerical variables and barplot for categorical ones.

print `signature(x = "MetaArray")`: prints the number of samples and genes in each dataset, followed by summarization of each clinical characteristic of the samples

show Same as `print`

as.list Function transforms a MetaArray object into a list, in which each slot is again a list of three slots: gene expression data matrix `GEDM`, clinical data `clinical`, name of the dataset `dataname`

Author(s)

Ivana Ihnatova

Examples

```
showClass("MetaArray")
```

`metagene` *Function to do extract row from list of data.frames*

Description

Function extracts one row (specified by number or name) from all data.frames of input list

Usage

```
metagene(x, results)
```

Arguments

`x`	number or name of row to be extracted
`results`	list of data frame (for example outputs of methods of meta-analysis where rows refer to genes or probesets)

Value

list, one slot refer to one data.frame

Author(s)

Ivana Ihnatova

Examples

```
A<-data.frame(x=rep(c(1,2,3),2),y=rep(c("a","b","c"),2))
B<-data.frame(x=rep(c(9,8,7),2),y=rep(c("x","y","z"),2))
res<-list(A=A,B=B)
metagene(2,res)
```

metaheat *Display Data as Heatmap*

Description

This function displays a matrix as a heatmap. It is based on function `heatmap_2` in the `Heatplus` package.

Usage

```
metaheat(x, Rowv = NA, Colv = NA, distfun = dist, hclustfun = hclust, na.rm = TRUE, do.dendro = c(TRUE,
```

Arguments

`x`	the numerical data matrix to be displayed
`Rowv`	either a dendrogram or a vector of reordering indexes for the rows, setting to `NA` suppresses re-ordering of rows
`Colv`	either a dendrogram or a vector of reordering indexes for the columns, setting to `NA` suppresses re-ordering of columns
`distfun`	function to compute the distances between rows and columns. Defaults to `dist`
`hclustfun`	function used to cluster rows and columns. Defaults to `hclust`
`na.rm`	logical indicating whther to remove NAs
`do.dendro`	logical vector of length two, indicating (in this order) whether to draw the row and column dendrograms
`legend`	integer between 1 and 4, indicating on which side of the plot the legend should be drawn: 1=bottom, 2=left, 3=above, 4=right
`legfrac`	fraction of the plot that is taken up by the legend; larger values correspond to smaller legends
`col`	the color scheme
`r.cex`	font size for row labels
`c.cex`	font size for column labels
`...`	extra arguments to `image`

Author(s)

Ivana Ihnatova

Examples

```
lists<-list(Method1=c("Gene_A", "Gene_V","Gene_S","Gene_C","Gene_U","Gene_D","Gene_E","Gene_G","Gene_W"),
  Method2=c("Gene_D","Gene_W","Gene_G","Gene_E","Gene_H","Gene_X"),
  Method3=c("Gene_L","Gene_K","Gene_J","Gene_M","Gene_V","Gene_T","Gene_R","Gene_U"))
A<-make.matrix(lists)
metaheat(A, legend=1, col=c(3,4))
```

`metaheat2` *Function to plot heatmap*

Description

Function is modification of function `heatmap.2` from package gplots

Usage

```
metaheat2(x, Rowv = TRUE, Colv = if (symm) "Rowv" else TRUE, distfun = dist, hclustfun = hclust, dendro
```

Arguments

`x`	numeric matrix of the values to be plotted.
`Rowv`	determines if and how the row dendrogram should be reordered. By default, it is TRUE, which implies dendrogram is computed and reordered based on row means. If NULL or FALSE, then no dendrogram is computed and no reordering is done. If a `dendrogram`, then it is used "as-is", ie without any reordering. If a vector of integers, then dendrogram is computed and reordered based on the order of the vector.
`Colv`	determines if and how the column dendrogram should be reordered. Has the options as the Rowv argument above and additionally when x is a square matrix, `Colv = "Rowv"` means that columns should be treated identically to the rows.
`distfun`	function used to compute the distance (dissimilarity) between both rows and columns. Defaults to `dist`.
`hclustfun`	function used to compute the hierarchical clustering when Rowv or Colv are not dendrograms. Defaults to `hclust`.
`dendrogram`	character string indicating whether to draw 'none', 'row', 'column' or 'both' dendrograms. Defaults to 'both'. However, if Rowv (or Colv) is FALSE or NULL and dendrogram is 'both', then a warning is issued and Rowv (or Colv) arguments are honoured.
`symm`	logical indicating if x should be treated symmetrically; can only be true when x is a square matrix.
`scale`	character indicating if the values should be centered and scaled in either the row direction or the column direction, or none. The default is `"row"` if `symm` false, and `"none"` otherwise.
`na.rm`	logical indicating whether NA's should be removed.
`revC`	logical indicating if the column order should be `rev`ersed for plotting, such that e.g., for the symmetric case, the symmetry axis is as usual.
`add.expr`	expression that will be evaluated after the call to `image`. Can be used to add components to the plot.
`breaks`	(optional) Either a numeric vector indicating the splitting points for binning x into colors, or a integer number of break points to be used, in which case the break points will be spaced equally between `min(x)` and `max(x)`.

`symbreaks`	Boolean indicating whether breaks should be made symmetric about 0. Defaults to `TRUE` if the data includes negative values, and to `FALSE` otherwise.
`col`	colors used for the image. Defaults to heat colors (`heat.colors`).
`colsep, rowsep, sepcolor`	(optional) vector of integers indicating which columns or rows should be separated from the preceding columns or rows by a narrow space of color `sepcolor`.
`sepwidth`	(optional) Vector of length 2 giving the width (colsep) or height (rowsep) the separator box drawn by colsep and rowsep as a function of the width (colsep) or height (rowsep) of a cell. Defaults to `c(0.05, 0.05)`
`cellnote`	(optional) matrix of character strings which will be placed within each color cell, e.g. p-value symbols.
`notecex`	(optional) numeric scaling factor for `cellnote` items.
`notecol`	(optional) character string specifying the color for `cellnote` text. Defaults to "green".
`na.color`	Color to use for missing value (`NA`). Defaults to the plot background color.
`trace`	character string indicating whether a solid "trace" line should be drawn across 'row's or down 'column's, 'both' or 'none'. The distance of the line from the center of each color-cell is proportional to the size of the measurement. Defaults to 'column'.
`tracecol`	character string giving the color for "trace" line. Defaults to "cyan".
`hline, vline, linecol`	Vector of values within cells where a horizontal or vertical dotted line should be drawn. The color of the line is controlled by linecol. Horizontal lines are only plotted if trace is 'row' or 'both'. Vertical lines are only drawn if trace 'column' or 'both'. hline and vline default to the median of the breaks, linecol defaults to the value of tracecol.
`margins`	numeric vector of length 2 containing the margins (see `par(mar= *)`) for column and row names, respectively.
`ColSideColors`	(optional) character vector of length `ncol(x)` containing the color names for a horizontal side bar that may be used to annotate the columns of `x`.
`RowSideColors`	(optional) character vector of length `nrow(x)` containing the color names for a vertical side bar that may be used to annotate the rows of `x`.
`cexRow, cexCol`	positive numbers, used as `cex.axis` in for the row or column axis labeling. The defaults currently only use number of rows or columns, respectively.
`labRow, labCol`	character vectors with row and column labels to use; these default to `rownames(x)` or `colnames(x)`, respectively.
`key`	logical indicating whether a color-key (legend) should be shown.
`keysize`	numeric value indicating the size of the key
`density.info`	character string indicating whether to superimpose a 'histogram', a 'density' plot, or no plot ('none') on the color-key.
`denscol`	character string giving the color for the density display specified by `density.info`, defaults to the same value as `tracecol`.

`symkey`	Boolean indicating whether the color key should be made symmetric about 0. Defaults to `TRUE` if the data includes negative values, and to `FALSE` otherwise.
`densadj`	Numeric scaling value for tuning the kernel width when a density plot is drawn on the color key. (See the `adjust` parameter for the `density` function for details.) Defaults to 0.25.
`main, xlab, ylab`	main, x- and y-axis titles; defaults to none.
`lmat, lhei, lwid`	visual layout: position matrix, column height, column width. See below for details
`legend.names`	character vector with labels of categories - used in legend
`discrete`	Logical, when `TRUE` boxes filled with the specified colors and names specified in `legend.names` are added as legend
`horiz`	Logical, when `TRUE` the legend is arranged horizontally
`...`	additional arguments passed on to `image`

Details

See function `heatmap.2` in gplots package for details

Author(s)

Ivana Ihnatova

`metalist.to.matrix` *Function to do convert list to matrix*

Description

Function converts list (output from `pvalcombination`, `EScombination`, `metaMA`) to matrix.)

Usage

```
metalist.to.matrix(list, genenames = NULL)
```

Arguments

`list`	output from `pvalcombination`, `EScombination`
`genenames`	vector of gene names in same order like in expression set for `pvalcombination`, can be `NULL` if the wrapper function `metaMA` was used.

Value

Matrix. Last columns contains test statistics (last slot from metalist). Other columns are binary vector indicating that the index of the gene was present in other slots of metalist.

Author(s)

Ivana Ihnatova

Examples

```
data(Singhdata)
pvalt<-pvalcombination(
  esets=Singhdata$esets,
  classes=Singhdata$classes,
  moderated = "t", BHth = 0.01)
xx<-metalist.to.matrix(pvalt,Singhdata$geneNames)
```

---

metaMA *Wrapper function for effect size or p-value combination methods*

Description

This is a wrapper function for effect size or p-value combination as implemented in metaMA package.

Usage

```
metaMA(data, varname, moderated = c("limma", "SMVar", "t")[1], BHth = 0.05, which = c("pval", "ES")[1]
```

Arguments

`data`	MetaArray object containing gene expression data matrices, clinical data matrices and a vector of data set names. The gene expression data matrices must have equal rownames
`varname`	Character String - name of one column in clinical data matrices to be used as class labels
`moderated`	Character - method to calculate the test statistic (or p-value) inside each study, one of: "limma", "SMVar" and "t"
`BHth`	Numeric - threshold for Benjamini Hochenberg adjusted p-values for selection of significant genes in meta-analysis
`which`	Character - choose `"pval"` for combination of p-values, or `"ES"` for effect sizes

Value

An object of class `"metaMA.res"`. It is a list where:

`Study1`	Vector of indices of differentially expressed genes in study 1. Similar names are given for the other individual studies.
`AllIndStudies`	Vector of indices of differentially expressed genes found by at least one of the individual studies.
`Meta`	Vector of indices of differentially expressed genes in the meta-analysis.
`TestStatistic`	Vector with test statistics for differential expression in the meta-analysis.

Author(s)

Ivana Ihnatova

References

Marot, G., Foulley, J.-L., Mayer, C.-D., Jaffrezic, F. Moderated effect size and p-value combinations for microarray meta-analyses.

See Also

pvalcombination, EScombination

Examples

```
data(ColonData)
pv<-metaMA(ColonData, "MSI", moderated = "t")
```

METRADISC *Wrapper function for METRADISC method*

Description

This is a wrapper function for meta-analytical method called METRADISC that perform all steps of the analysis and return all the outputs in one line.

Usage

```
METRADISC(data, varname, nperm = 1000)
```

Arguments

`data`	MetaArray object
`varname`	Character String - name of one column in clinical data matrices to be used as class labels
`nperm`	Number of permutations for Monte Carlo permutation test, at least 1000 is suggested

Value

An object of class `METRADISC.res` containing

`ranks`	Ranks of the genes in each dataset
`RQ`	Average rank (R) and measure of heterogeneity (Q) for each gene
`MCtest`	Four p-values (for low and high R and low and high Q) for each gene as provided after Monte Carlo permutation test

Author(s)

Ivana Ihnatova

References

Zintzaras, E., Ioannidis, J.P.A 2008 Meta-analysis for ranked discovery datasets: Theoretical framework and empirical demonstration for microarrays, Computational Biology and Chemistry 32, 39-47

Examples

```
data(ColonData)
m <- METRADISC(ColonData, "MSI", 5)
```

`patternMatch` *Function to count soft pattern matches*

Description

Funtion counts number of observed soft matches in meta-analysis

Usage

```
patternMatch(X.discret, unique.pat)
```

Arguments

`X.discret`	Binary matrix, with rows corresponding to genes, columns to studies and 1 to selected (significant) genes in studies
`unique.pat`	Vector of binary strings - patterns

Value

Numeric vector of number of soft pattern matches for each pattern.

Author(s)

Code provided by Xinan Yang <xnyang@seu.edu.cn>

References

Yang, X., Bentink, S. a Spang, R. 2005, Detecting Common Gene Expression Patterns in Multiple Cancer Outcome Entities, Biomedical Microdevices, vol.7:3

Examples

```
A<-matrix(c(1,0,0,1,0,1,0,1,1,0,1,0,1,0,1), ncol=3, nrow=10)
uni<-c("011","101","110","111")
patternMatch(A,uni)
```

---

patternMatch.strong *Function to count strong pattern matches*

Description

Funtion counts number of observed strong matches in meta-analysis

Usage

```
patternMatch.strong(X.discret, unique.pat)
```

Arguments

`X.discret`	Binary matrix, with rows corresponding to genes, columns to studies and 1 to selected (significant) genes in studies
`unique.pat`	Vector of binary strings - patterns

Value

Numeric vector of number of strong pattern matches for each pattern.

Author(s)

Code provided by Xinan Yang <xnyang@seu.edu.cn>

References

Yang, X., Bentink, S. a Spang, R. 2005, Detecting Common Gene Expression Patterns in Multiple Cancer Outcome Entities, Biomedical Microdevices, vol.7:3

Examples

```
A<-matrix(c(1,0,0,1,0,1,0,1,1,0,1,0,1,0,1), ncol=3, nrow=10)
uni<-c("011","101","110","111")
patternMatch.strong(A,uni)
```

`patternmatrix` *Function to split binary vectors to matrix.*

Description

Funtion takes vector of binary strings (0,1) and returns matrix with strings split.)

Usage

```
patternmatrix(unipattern, n.entity)
```

Arguments

`unipattern`	Vector of binary strings
`n.entity`	Length of strings, number of studies in original application

Details

Originally part of MAP-Matches implementation

Value

Binary matrix with rows corresponding to input strings.

Author(s)

Code provided by Xinan Yang <xnyang@seu.edu.cn>

References

Yang, X., Bentink, S. a Spang, R. 2005, Detecting Common Gene Expression Patterns in Multiple Cancer Outcome Entities, Biomedical Microdevices, vol.7:3

Examples

```
uni<-c("0101","1100","0011","0100")
patternmatrix(unipattern=uni,n.entity=4)
```

`patternToString` *Function to convert rows of a matrix to strings*

Description

Function takes a matrix and converts rows of it to strings - One string per row.

Usage

```
patternToString(X.discret)
```

Arguments

`X.discret`	Matrix

Details

Originally part of MAP-Matches implementation

Value

Matrix with same number of rows as input, but with rows converted to strings.

Author(s)

Code provided by Xinan Yang <xnyang@seu.edu.cn>

References

Yang, X., Bentink, S. a Spang, R. 2005, Detecting Common Gene Expression Patterns in Multiple Cancer Outcome Entities, Biomedical Microdevices, vol.7:3

Examples

```
A<-matrix(c(0,1,0,0,1,1), ncol=4, nrow=5)
patternToString(A)
#another example
uni<-c("0101","1100","0011","0100")
A<-patternmatrix(uni,3)
patternToString(A)
```

performSOGL *Function to perform analysis using Similarity of Ordered Gene Lists*

Description

This a wrapper function to perform all steps designed in Similarity of Ordered Gene Lists.

Usage

```
performSOGL(data, varname, test, B, which = c("score", "empirical"), min.weight = 1e-05, two.sided = T
```

Arguments

data	MetaArray object, the rownames in the gene expression data matrices must be equal
varname	A string indicating which column of clinical data matrices should be used to compute test statistic. Same column is used in all datasets.
test	`"FCH"` for fold change (function `fold.change`) or `"T"` for T-test (function `meta.test`)
B	Number of permuatations
which	if `"empirical"` then empirical confidence intervals of number of overlapping genes are also provided, if `"score"` only random and subsampled scores necessary for tunning alpha parameter are calculated
min.weight	Minimal weight for score calculation
two.sided	if `TRUE` both top and bottom of the ordered gene lists are considered, if `FALSE` only top ones
percent	Percentage (Numeric between `0` and `1`) of the score for genes selection

Value

Object of class SOGLresult, it is a list containig:

ordering	Ordered Gene Lists as a data.frame where columns refer to datasets
alpha.selected	Selected value of alpha parameter
alpha.considered	Vector of alpha considered for selection
pAUC	pAUC values related to all alphas considered
random	Random scores (permutations of class labels)
subsample	Scores after subsampling from each class and dataset
emp.ci	Empirical confidence intervals for number of overlapping genes
common.genes	Vector of number of overlapping genes
score	Observed similarity score
significance	Significance of the observed score in form of p-value
genes	Genes that account for observed similarity score
all.genes	Names of the all genes in the analysis

Author(s)

Ivana Ihnatova

References

Yang, X., Bentink, S., Scheid, S. Spang, R., Similarities of ordered gene lists, 2005

`plot.SOGLresult` *Function to plot an object of class SOGLresult*

Description

Function draws three plots presented for results of meta-analysis by Similarity of Ordered Gene List method

Usage

```
## S3 method for class 'SOGLresult'
plot(x, which, ...)
```

Arguments

`x`	SOGLresult object, provided by function `performSOGL`
`which`	Character indicator which plot has to be drawn, see Details
`...`	arguments to `plot` function

Details

If `which="alpha selection"` Considered alphas and corresponding pAUC are plotted. Red vertical line signs selected alpha. If `which="density"`Estimated density of random (in black) and subsampled score (in red) If `which="empirical CI"`Empirical confidence intervals and observed number of overlapping genes

Author(s)

Ivana Ihnatova

References

Yang, X., Bentink, S., Scheid, S. Spang, R., Similarities of ordered gene lists, 2005

plotES *Function to do plots in combination of effect size method*

Description

Function plots several characteristics examined in meta-analysis with combination effect size method.

Usage

```
plotES(theScores,ScoresFDR,num.studies, legend.names, colors, which)
```

Arguments

`theScores`	Output from function `zScores`
`ScoresFDR`	Output from function `zScoreFDR`
`num.studies`	number of studies involved in meta-analysis
`legend.names`	vector of names of studies, the first one should be "Combined Set"
`colors`	vector of colors used in plots, its length must be 1 + number of studies
`which`	subset from 1,2,3: 1 for plot of the fraction of the genes that have a higher effect size than the threshold for the combined Z-score, but not for any of the data set specific Z-scores, 2 for plot of the number of genes and the corresponding FDR for the two sided situation and 3 for plot of the number of genes that are below a given threshold for the FDR

Author(s)

Ivana Ihnatova

See Also

`zScores`, `zScoreFDR`

plotgene *Function to visuaze change in expression of one gene*

Description

Various methods for meta-analysis provide different outputs. Function takes an output from function `metagene` as input and draws a plot.

Usage

```
plotgene(gene, datalabels=NULL, type, col=c("green", "red"),cex=c(0.7),sig=0.05)
```

Arguments

`gene`	A list, output from function `metagene`
`datalabels`	A character vector, names of the data sets and for meta-analysis results. If `NULL`, dummy names `Study1`, `Study2`, `Study3`, ..., `Meta` are created.
`type`	A numeric vector indicating which function the slots of gene come from. It it is not necessary, if the slots come from the results of the wrapper functions.
`col`	colors for unsignificant/significant
`cex`	Font size for labels in unsignificant/significant part of the chart
`sig`	Significance threshold for p-values graph

Details

Function `plotgene2` is based on traditional graphics, wheras function `plotgene` on grid.

For `type` please use:

- 0 for functions: `pvalcombination`, `pvalcombination.paired`, `EScombination` or `EScombination.paired`, `metaMA`
- 1 for function `ES.GeneMeta`
- 2 for function `zScores`
- 3 for function `ScoresFDR`
- 4 for function `performSOGL`
- 5 for function `topGene` or `RankProduct`
- 6 for function `posterior.mean`
- 7 for function `MAP.genes` or `MAP.Matches`
- 8 for function `MC`
- 9 for function `compute.RQ`
- 10 for function `METRADISC`

Author(s)

Ivana Ihnatova

`plotgene2` *Function to visuaze change in expression of one gene*

Description

Various methods for meta-analysis provide different outputs. Function takes an output from function `metagene` as input and draws a plot.

Usage

```
plotgene2(gene, datalabels, type)
```

Arguments

`gene`	A list, output from function `metagene`
`datalabels`	A character vector, names of the data sets and for meta-analysis results.
`type`	A numeric vector indicating which function the slots of gene come from. It it is not necessary, if the slots come from the results of the wrapper functions.

Details

Function `plotgene2` is based on traditional graphics, wheras function `plotgene` on grid.

For `type` please use:

- 0 for functions: `pvalcombination`, `pvalcombination.paired`, `EScombination` or `EScombination.paired`, `metaMA`
- 1 for function `ES.GeneMeta`
- 2 for function `zScores`
- 3 for function `ScoresFDR`
- 4 for function `performSOGL`
- 5 for function `topGene` or `RankProduct`
- 6 for function `posterior.mean`
- 7 for function `MAP.genes` or `MAP.Matches`
- 8 for function `MC`
- 9 for function `compute.RQ`
- 10 for function `METRADISC`

Author(s)

Ivana Ihnatova

`plotpattern` *Function to do plot signifficance of Meta-Analysis Patterns*

Description

Function plots signifficance of Meta-Analysis Patterns

Usage

```
plotpattern(intx, method)
```

Arguments

`intx`	A data frame with rows refering to Meta-Analysis Patterns and columns (from 5th to 8th) to signifficance of observed pattern matches
`method`	Either number 1 or 2, otherwise no plot is provided. If 1 a line plot is made. If 2 a form of scatterplot is drawn.

Author(s)

Ivana Ihnatova

References

Yang, X., Bentink, S. a Spang, R. 2005, Detecting Common Gene Expression Patterns in Multiple Cancer Outcome Entities, Biomedical Microdevices

---

`plotQvsChi` *Function to plot quantiles of Cochran's Q statistic and Chi-square quantiles.*

Description

Function plots quantiles of Cochran's Q statistic and Chi-square quantiles as a scatter plot with diagonal line. Plot can help to decide between random-effect and fixed-effect model. It is a wrapper function to provide a complete plot.

Usage

```
plotQvsChi(Q, num.studies)
```

Arguments

`Q`	A vector of Cochran's Q statistic used to test between-study variability.
`num.studies`	Number of studies involved in meta-analysis.

Author(s)

Ivana Ihnatova

`posterior.mean` *Function to calculate posterior mean differential expression*

Description

Function calculates posterior mean differential expression in form of Z-score and its p-value as described in Wang et al., 2004.

Usage

```
posterior.mean(data, varname, nsamp, permute = 0)
```

Arguments

`data`	An MetaArray object
`varname`	A string indicating which column of clinical data matrices should be used to compute test statistic. Same column is used in all datasets.
`nsamp`	Number of samples. It is suggested to use same number of samples in each class and dataset.
`permute`	If permute is 0, weighted Z-score will be referenced to standard normal distribution for two-sided p-value. Otherwise, columns of all datasets (each dataset separately) will be shuffled at random, from which a permutation distribution of Z-scores are formed and Z-scores are referenced to this distribution.

Details

The main idea of this method is that one can use data from one study to con- struct a prior distribution of differential expression and thus utilize the posterior mean differential expression, weighted by variances, whose distribution is stan- dard normal distribution due to classic Bayesian probability calculation. It is based on assumption that gene expression is normally distributed and that we can estimate the standard deviation of this distribution by pooling together all genes with similar levels of mean expression.

Value

Object of class `posterior.mean`. It is a data frame, where the first column contain Z-scores, the second p-values, rows refer to genes.

Author(s)

Ivana Ihnatova

References

Wang, J., Coombes, K. R., Highsmith, W. E., Keating, M. J. a Abruzzo, L. V. 2004, Differences in gene expression between B-cell chronic lymphocytic leukemia and normal B cells: a meta-analysis of three microarray studies

Examples

```
data(Singhdata)

cl1<-as.data.frame(Singhdata$classes[[1]])
names(cl1)<-"classlab"
cl2<-as.data.frame(Singhdata$classes[[2]])
names(cl2)<-"classlab"
cl3<-as.data.frame(Singhdata$classes[[3]])
names(cl3)<-"classlab"
rownames(Singhdata$esets[[1]])<-Singhdata$geneNames
rownames(Singhdata$esets[[2]])<-Singhdata$geneNames
rownames(Singhdata$esets[[3]])<-Singhdata$geneNames
```

```
data<-new("MetaArray", GEDM=list(Singhdata$esets[[1]], Singhdata$esets[[2]], Singhdata$esets[[3]]),
clinical=list(cl1, cl2, cl3), datanames=c("dataset1", "dataset2", "dataset3"))

pm<-posterior.mean(data, "classlab", 4)
head(pm)
```

`prelimScore` *Function compute preliminary Similarity Score for Ordered Gene Lists*

Description

Function computes preliminary Similarity Score as defined in Yang, 2005.

Usage

```
prelimScore(ordering, alpha, min.weight = 1e-05, two.sided = TRUE)
```

Arguments

`ordering`	Data frame, where columns refer to ordered gene list from one study
`alpha`	Numeric parameter used in weights `exp(-alpha*n)`
`min.weight`	minimal weight to be counted
`two.sided`	if `TRUE` both top and bottom of the `ordering` considered, if `FALSE` only top positions are considered

Value

Similarity Score

Author(s)

Ivana Ihnatova

References

Yang, X., Bentink, S., Scheid, S. Spang, R., Similarities of ordered gene lists, 2005

Examples

```
genes<-paste("Gene", 1:100)
O<-cbind(sample(genes), sample(genes), sample(genes))
prelimScore(O, 0.1)
```

`prepareData` *Function to prepare data*

Description

Function prepares data as part of `RandomScore` function

Usage

```
prepareData(j, data, varname, p, type)
```

Arguments

`j`	Permutation
`data`	MetaArray object - original dataset
`varname`	String indicating which column of clinical data matrices should be used to compute test statistic. Same column is used in all datasets.
`p`	Permutation of class labels or subsample
`type`	`1` for permutation of class labels, `2` for subsamples

Value

MetaArray object

Author(s)

Ivana Ihnatova

`preparePermutations` *Function to prepare permutation and subsamples*

Description

Function prepares permutations of class labels and subsamples from expression data

Usage

```
preparePermutations(id, B, sample.ratio = 0.8)
```

Arguments

`id`	Binary vector (0's and 1's) of class labels to be permuted and subsampled
`B`	number of premutations
`sample.ratio`	ratio for subsampling, default `0.8` means 80% of samples from each group is selected

Value

A list

`yperm`	Permutation - vectors of 0's and 1's shuffeled
`ysubs`	Subsamples - numeric vectors indicating which samples should be selected

Author(s)

Ivana Ihnatova, Claudio Lottaz

`probs.to.matrix` *Function to convert list to matrix*

Description

Function converts list to binary matrix

Usage

```
probs.to.matrix(probs, genenames)
```

Arguments

`probs`	list of vectors of gene names/ character strings
`genenames`	vector of all gene names in analysis / all strings to be considered

Value

matrix in which rows refer to genes (character strings) and columns to slots of input list

Author(s)

Ivana Ihnatova

See Also

`metalist.to.matrix`

Examples

```
lists<-list(Method1=c("Gene_A", "Gene_V","Gene_S","Gene_C","Gene_U","Gene_D","Gene_E","Gene_G","Gene_W"),
  Method2=c("Gene_D","Gene_W","Gene_G","Gene_E","Gene_H","Gene_X"),
  Method3=c("Gene_L","Gene_K","Gene_J","Gene_M","Gene_V","Gene_T","Gene_R","Gene_U"))
genes<-c("Gene_A", "Gene_V", "Gene_S", "Gene_C", "Gene_U", "Gene_D", "Gene_E", "Gene_G",
 "Gene_W", "Gene_H", "Gene_X", "Gene_L", "Gene_K", "Gene_J", "Gene_M", "Gene_T",
 "Gene_R")
PM<-probs.to.matrix(lists,genes)
PM
```

RandomScore *Function to do compute random and subsampled similarity score*

Description

Function computes random and subsampled similarity score in order to select appropriate tunning parameter alpha for weights in preliminary simmilarity score

Usage

```
RandomScore(data, varname, B, alpha, test, which = c("random", "empirical", "subsample"), two.sided =
```

Arguments

`data`	MetaArray object
`varname`	String indicating which column of clinical data should be used as class labels, same on all data set
`B`	Number of permutation
`alpha`	Vector of alphas considered (can be returned by `computeAlpha`) or selected manually
`test`	`"FCH"` for fold change (function `fold.change`) or `"T"` for T-test (function `meta.test`)
`which`	`"random"` for random score (permutation of class labels), `"subsample"` for subsampled score, `"empirical"` for empirical confidence intervals of overalaping genes, vector of several is also possible
`two.sided`	if `TRUE` both top and bottom of the `ordering` considered, if `FALSE` only top positions are considered

Value

A list

`random`	Random similarity score
`empirical`	Empirical confidence intervals for overlaping genes
`subsample`	Subsampled score

Author(s)

Ivana Ihnatova

rank.genes *Rank genes*

Description

Assigna ranks to gene names according to p-value and sign of test statistics

Usage

```
rank.genes(T, p)
```

Arguments

T	vector of test statistics with gene names in `names`
p	vector of p-values with gene names in `names`

Value

Data frame with ranks of gene names

Author(s)

Ivana Ihnatova

References

Zintzaras, E., Ioannidis, J.P.A 2008 Meta-analysis for ranked discovery datasets: Theoretical framework and empirical demonstration for microarrays, Computational Biology and Chemistry 32, 39-47

See Also

compute.RQ

Examples

```
## Not run:
data(Singhdata)

#compute T-statistics and P-value
p1<-apply(Singhdata$esets[[1]],1,function(x) {t=t.test(x~Singhdata$classes[[1]], alternative="two.sided"); retu
p2<-apply(Singhdata$esets[[2]],1,function(x) {t=t.test(x~Singhdata$classes[[2]], alternative="two.sided"); retu
p3<-apply(Singhdata$esets[[3]],1,function(x) {t=t.test(x~Singhdata$classes[[3]], alternative="two.sided"); retu
T1<-apply(Singhdata$esets[[1]],1,function(x) {t=t.test(x~Singhdata$classes[[1]], alternative="two.sided"); retu
T2<-apply(Singhdata$esets[[2]],1,function(x) {t=t.test(x~Singhdata$classes[[2]], alternative="two.sided"); retu
T3<-apply(Singhdata$esets[[3]],1,function(x) {t=t.test(x~Singhdata$classes[[3]], alternative="two.sided"); retu

# Rank genes
rank1<-rank.genes(T1,p1)
```

```
rank2<-rank.genes(T2,p2)
rank3<-rank.genes(T3,p3)

## End(Not run)
```

`rank.genes.adv` *Function to rank genes*

Description

A wrapper function to `rank.genes`. This function provides a data frame of ranks, where rows correspond to genes and columns to data sets.

Usage

```
rank.genes.adv(testp)
```

Arguments

`testp`	A list with two slots: `test` and `p`, where `test` is data frame of T-statistics and `p` is data frame of p-values. In both rows refer to genes and columns to data sets. It is output from `meta.test` function.

Value

A data frame of ranks, where rows correspond to genes and columns to data sets.

Author(s)

Ivana Ihnatova

See Also

`meta.test`, `rank.genes`

Examples

```
 data(Singhdata)

cl1<-as.data.frame(Singhdata$classes[[1]]+1)
names(cl1)<-"classlab"
cl2<-as.data.frame(Singhdata$classes[[2]]+1)
names(cl2)<-"classlab"
cl3<-as.data.frame(Singhdata$classes[[3]]+1)
names(cl3)<-"classlab"
rownames(Singhdata$esets[[1]])<-Singhdata$geneNames
rownames(Singhdata$esets[[2]])<-Singhdata$geneNames
rownames(Singhdata$esets[[3]])<-Singhdata$geneNames
```

```
data<-new("MetaArray", GEDM=list(Singhdata$esets[[1]], Singhdata$esets[[2]], Singhdata$esets[[3]]),
clinical=list(cl1, cl2, cl3), datanames=c("dataset1", "dataset2", "dataset3"))

m<-meta.test(data,"classlab")
rank.genes.adv(m)
```

`RankProduct` *Wrapper function for RankProduct method*

Description

This is a wrapper function for perfoming meta-analysis using Rank Product method.

Usage

```
RankProduct(data, varname, num.perm = 100, logged = TRUE, na.rm = FALSE, gene.names = NULL, plot = FALS
```

Arguments

`data`	MetaArray object
`varname`	Character String - name of one column in clinical data matrices to be used as class labels, factors are turned into a numeric vector by `as.numeric()-1`)
`num.perm`	Number of permutations
`logged`	Logical - indicating whether data are on log-scale
`na.rm`	Logical - if `FALSE` (default), the NA value will not be used in computing rank. If `TRUE` the missing values will be replaced by the genewise mean of the non-missing values. Gene will all value missing will be assigned "NA"
`gene.names`	Character vector - gene names to be be attached to the estimated percentage of false prediction (pfp)
`plot`	Logical - if `TRUE` a plot of the estimated pfp verse the rank of each gene is drawn
`rand`	Numeric - a seed for random number generator
`cutoff`	Numeric - p-value for selection of significant genes

Value

Object of class `RankProduct.res` containing outputs from functions: `RPadvance` and `topGene`. 'Class1' refers to the first level of the used class labels, 'Class2' to the second one.

Author(s)

Ivana Ihnatova

References

Breitling, R., Armengaud, P., Amtmann, A., and Herzyk, P.(2004) Rank Products: A simple, yet powerful, new method to detect differentially regulated genes in replicated microarray experiments, FEBS Letter, 57383-92

Examples

```
## Not run:
data(ColonData)
rp<-RankProduct(ColonData, "MSI", num.perm=10)

## End(Not run)
```

---

ratio *Function to calculate the ratio of co-significant: expected/observed*

Description

Function to calculates the ratio of co-significant: expected/observed of strong and soft pattern matches

Usage

```
ratio(X.discret)
```

Arguments

`X.discret`	Matrix of 0 and 1. Rows correspond to genes, columns to studies (comparisons). 1 means that T statistic for the gene in the study was higher than selected threshold, otherwise 0 is used.

Details

Calculation is part of MAP-Matches methods. See References for details\

Value

A list which contains

`n`	Number of selected genes in each comparison
`X.string`	Patterns in observed in data
`p.strong`	Co-significance of strong pattern match, probability of observing strong pattern match in data?
`p.soft`	Co-significance of soft pattern match, probability of observing strong pattern match in data?

Author(s)

Codes provided by Xinan Yang <xnyang@seu.edu.cn>

References

Yang, X., Bentink, S. a Spang, R. 2005, Detecting Common Gene Expression Patterns in Multiple Cancer Outcome Entities, Biomedical Microdevices, vol.7:3

Examples

```
A<-matrix(c(1,0,0,1,0,1,0,1,1,0,1,0,1,0,1), ncol=3, nrow=10)
ratio(A)
```

sd.filter *Microarray probes filtering*

Description

Function to filter microarray probes according to standard deviation

Usage

```
sd.filter(data, cutoff = 0.5)
```

Arguments

`data`	expression matrix with probes in rows and samples in columns
`cutoff`	cutoff value for filtering

Value

Expression matrix, probes with SD below cutoff are filtered out.

Author(s)

Ivana Ihnatova

Examples

```
data(Singhdata)
data<-Singhdata$esets[[1]][1:1000,]
data.filtered<-sd.filter(data)
head(data.filtered)
```

selectAlpha *Function to select the most optimal alpha parameter*

Description

Function selects the most optimal value of alpha parameter according to pAUC. For each of possible alphas the pAUC is computed as a measure of the separabilty of two distributions of similarity score: random and subsampled (prepared by function `RandomScore`. Alpha with maximal pAUC is selected.

Usage

```
selectAlpha(alpha, subsample, random)
```

Arguments

`alpha`	Vector of possible alphas
`subsample`	Similarity scores after subsampling
`random`	Similarity scores after permuting class labels

Value

A list:

`alpha`	selected value of alpha
`pAUC`	pAUC for all alphas achivied

Author(s)

Ivana Ihnatova

See Also

`RandomScore`

selectClass *Function to select class labels from MetaArray object*

Description

Function selects one column from each clinical data matrix and binds them into a list object

Usage

```
selectClass(x, varname, type)
```

Arguments

`x`	MetaArray object
`varname`	Character String specifying which column of clinical data should be selected
`type`	if `factor` then factor vector is returned , if `binary` then vector of 1's and 0' is returned as class labels

Value

A list where each slot refers to one clinical data matrix (study) and contains selected class labels of the samples.

Note

Such a class labels extraction is necessary for some methods of meta-analysis of microarray

Author(s)

Ivana Ihnatova

Examples

```
data(Singhdata)

cl1<-as.data.frame(Singhdata$classes[[1]])
names(cl1)<-"classlab"
cl2<-as.data.frame(Singhdata$classes[[2]])
names(cl2)<-"classlab"
cl3<-as.data.frame(Singhdata$classes[[3]])
names(cl3)<-"classlab"
rownames(Singhdata$esets[[1]])<-Singhdata$geneNames
rownames(Singhdata$esets[[2]])<-Singhdata$geneNames
rownames(Singhdata$esets[[3]])<-Singhdata$geneNames

dataset<-new("MetaArray", GEDM=list(Singhdata$esets[[1]], Singhdata$esets[[2]], Singhdata$esets[[3]]),
clinical=list(cl1, cl2, cl3), datanames=c("dataset1", "dataset2", "dataset3"))

selectClass(dataset, "classlab", "factor")
selectClass(dataset, "classlab", "binary")
```

`selectGenes` *Function to select genes that account for Similarity score*

Description

Genes that account for e.g 95% of Similarity score are returned.

Usage

```
selectGenes(ordering, alpha, percent, min.weight = 1e-05, two.sided = TRUE)
```

Arguments

`ordering`	Ordered gene lists as data.frame or matrix, each column refer to one study
`alpha`	Selected alpha parameter for Similarity score
`percent`	Percentage (Numeric between `0` and `1`) of the score
`min.weight`	minimal weight to be counted
`two.sided`	if `TRUE` both top and bottom of the `ordering` considered, if `FALSE` only top positions are considered

Value

Vector of genes

Author(s)

Ivana Ihnatova

Examples

```
genes<-paste("Gene", 1:1000)

O<-cbind(c(sample(genes[1:200]),sample(genes[201:1000])),
         c(sample(genes[1:200]),sample(genes[201:1000])),
         c(sample(genes[1:200]),sample(genes[201:1000]))
)
alph<-computeAlpha(100,ngenes=1000)
selectGenes(O, alph, 0.95)
```

`sigScore` *Function to calculate signifficance of similarity score*

Description

Function calculates empirical signifficance of similarity score by means of random permutation of ordered gene lists, computing the similarity scores and comparing them to the value observed in original ordering of the genes.

Usage

```
sigScore(ranking, alpha, B, min.weight = 1e-05, two.sided = TRUE)
```

Arguments

`ranking`	Ordered gene lists as data.frame or matrix, each column refer to one study
`alpha`	Selected alpha parameter for Similarity score
`B`	Number of permutation
`min.weight`	Minimal weight for similarity score calculation
`two.sided`	if `TRUE` both top and bottom of the `ordering` considered, if `FALSE` only top positions are considered

Value

Signifficance of similarity score in form of p-value

Author(s)

Ivana Ihnatova

Examples

```
## Not run:
genes<-paste("Gene", 1:100)
O<-cbind(sample(genes), sample(genes), sample(genes))
sigScore(O, 0.0001, 100)

## End(Not run)
```

`T.select` *Function to help with selection of threshold for T-statistics*

Description

Function calculates quantiles of T-statistics to help with selection of threshold for it as part of MAP-Matches method.

Usage

```
T.select(stat, fig = TRUE)
```

Arguments

`stat`	Vector of T-statistics
`fig`	If `TRUE` a plot of quantiles and sequence from 0.97 to 0.98 is provided.

Value

A vector of T-statistics quantiles.

Author(s)

Ivana Ihnatova

Examples

```
## The function is currently defined as
function(stat,fig=TRUE)
{
 quan <- quantile(abs(stat),seq(0.97,0.98,.0001))
 x <- seq(0.97,0.98,1e-04)
if (fig) {
 plot(x,quan,type="b",xlab="percent",ylab="t")
 z <- lm(quan ~ x)
 abline(z,col="red")
 points(0.9787,quan["97.87%"],pch=19,cex=1.5,col="red")
 }
return(quan)
  }
```

`test.group.shuffle` *Function to do compute test statistic iterativelly*

Description

Function computes test statistic with random assignment of group labels to samples in each iteration. It binds results to one matrix. Finaly it multiplies values of test statistics by -1. It saves a file necessary in MAP-Matches method.

Usage

```
test.group.shuffle(x,  varname, minSampleNum = 3, method = "t",B=1000)
```

Arguments

`x`	MetaArray object
`varname`	String indicating the column of clinical data matrices definig groups
`minSampleNum`	Minimal number of samples required for test statistic
`method`	Type of test as in `mt.teststat`
`B`	Number of iterations

Value

matrix of test statistics (with random group assignment and multiplied by -1)

Author(s)

Ivana Ihnatova

References

Yang, X., Bentink, S. a Spang, R. 2005, Detecting Common Gene Expression Patterns in Multiple Cancer Outcome Entities, Biomedical Microdevices, vol.7:3

See Also

`entitybuild2`

`VennMapper` *Wrapper function for VennMapping*

Description

This is a wrapper function for meta-analysis using VennMapping method. It performs all necessary steps and provides all available outputs.

Usage

```
VennMapper(data, varname, cutoff)
```

Arguments

`data`	MetaArray object
`varname`	Character String - name of one column in clinical data matrices to be used as class labels
`cutoff`	Numeric - cutoff for selection of genes according to their fold-change in log2-scale. e.g. 1 equals to two-fold expression change

Value

An object of class `VennMapper.res` containing

`conting.tab`	A contingency table with numbers of overlapping genes in pairs of the datasets
`z.score`	A table of z-scores describing the significance of overlap in pairs of the datasets
`genes`	A table of gene names that overlap in pairs of the datasets

Author(s)

Ivana Ihnatova

References

Smid, M., Dorssers, L. C. J. and Jenster, G. 2003, Venn Mapping: clustering of heterologous microarray data based on the number of co-occurring differentially expressed genes, Bioinformatics, vol. 19 no. 16 2003

Examples

```
data(ColonData)
vm<-VennMapper(ColonData, "MSI", 1)
```

Z *Function to compute Z-statistics in contingency table*

Description

Function to compute Z-statistics in contingency table as in VennMapper program.

Usage

```
Z(lists, n.genes)
```

Arguments

`lists`	list of vectors. Each vector refers to a method and contains names of significant genes
`n.genes`	Number of genes in meta-analysis (number of genes on microarray slide)

Details

Simmilar to `conting.tab` and `gene.list`, but provides different output

Value

Matrix of Z-statistics as defined in see references.

Author(s)

Ivana Ihnatova

References

Smid, M., Dorssers, L. C. J. and Jenster, G. 2003, Venn Mapping: clustering of heterologous microarray data based on the number of co-occurring differentially expressed genes, Bioinformatics, vol. 19 no. 16 2003

See Also

`conting.tab`, `gene.list`

Examples

```
lists<-list(Method1=c("Gene_A", "Gene_V","Gene_S","Gene_C","Gene_U","Gene_D","Gene_E","Gene_G","Gene_W"),
  Method2=c("Gene_D","Gene_W","Gene_G","Gene_E","Gene_H","Gene_X"),
  Method3=c("Gene_L","Gene_K","Gene_J","Gene_M","Gene_V","Gene_T","Gene_R","Gene_U"))
z<-Z(lists, n.genes=10000)
```

zScores	*Function for Meta-analysis of gene expression data*

Description

Functions for computing zScores for FEM and REM and computing FDR. This are modification of functions found in GeneMeta package.

Usage

```
zScores(esets, classes, useREM=TRUE, CombineExp=1:length(esets))
zScorePermuted(esets, classes, useREM=TRUE, CombineExp=1:length(esets))
zScoreFDR(esets, classes, useREM=TRUE, nperm=1000, CombineExp=1:length(esets))
multExpFDR(theScores, thePermScores, type="pos")
```

Arguments

`esets`	A `list` of `matrices`, one expression set per experiment. All experiments must have the same variables(genes).
`classes`	A `list` of class memberships, one per experiment. Each `list` can only contain 2 levels.
`useREM`	A `logical` value indicating whether or not to use a REM, `TRUE`, or a FEM, `FALSE`, for combining the z scores.
`theScores`	A `vector` of scores (e.g. t-statistics or z scores)
`thePermScores`	A `vector` of permuted scores (e.g. t-statistics or z scores)
`type`	`"pos"`, `"neg"` or `"two.sided"`
`nperm`	number of permutations to calculate the FDR
`CombineExp`	A `vector` of `integer`- which experiments should be combined-default:all experiments

Details

The function `zScores` implements the approach of Choi et al. for MetaArray. The function `zScorePermuted` applies zScore to a single permutation of the class labels. The function `zScoreFDR` computes a FDR for each gene, both for each single experiment and for the combined experiment. The FDR is calculated as described in Choi et al. Up to now ties in the zscores are not taken into account in the calculation. The function might produce incorrect results in that case. The function also computes zScores, both for the combines experiment and for each single experiment.

Value

A `matrix` with one row for each probe(set) and the following columns:

`zSco_Ex_`	For each single experiment the standardized mean difference, Effect_Ex_, divided by the estimated standard deviation, the square root of the EffectVar_Ex_ column.

`MUvals`	The combined standardized mean difference (using a FEM or REM)
`MUsds`	The standard deviation of the MUvals.
`zSco`	The z statistic - the MUvals divided by their standard deviations, MUsds.
`Qvals`	Cochran's Q statistic for each gene.
`df`	The degree of freedom for the Chi-square distribution. This is equal to the number of combined experiments minus one.
`Qpvalues`	The probability that a Chi-square random variable, with df degrees of freedom) has a higher value than the value from the Q statistic.
`Chisq`	The probability that a Chi-square random variate (with 1 degree of freedom) has a higher value than the value of zSco^2.
`Effect_Ex_`	The standardized mean difference for each single experiment.
`EffectVar_Ex_`	The variance of the standardized mean difference for each single experiment.

Note that the three column names that end in an underscore are replicated, once for each experiment that is being analyzed.

Author(s)

M. Ruschhaupt (original function), I. Ihnatova (modification)

References

Choi et al, Combining multiple microarray studies and modeling interstudy variation. Bioinformatics, 2003, i84-i90.

Examples

```
data(ColonData)
esets <- GEDM(ColonData)
classes <- selectClass(ColonData, "MSI", "binary")
theScores <- zScores(esets, classes, useREM = FALSE)
```

Index