Package 'gCMAP'

October 9, 2013

Type Package

Title Tools for Connectivity Map-like analyses

Version 1.4.1

Date 2013-07-30

Depends GSEABase, limma (>= 3.15.14)

Imports Biobase, BiocGenerics, methods, GSEAlm, Category, bigmemory,bigmemoryExtras (>= 1.1.2), Matrix (>= 1.0.9), parallel,annotate, genefilter, AnnotationDbi

Suggests DESeq, KEGG.db, reactome.db, RUnit, GO.db, mgsa

Author Thomas Sandmann <sandmann.thomas@gene.com>, Richard Bourgon <bourgon.richard@gene.com> and Sarah Kummerfeld <kummerfeld.sarah@gene.com>

Maintainer Thomas Sandmann <sandmann.thomas@gene.com>

Description The gCMAP package provides a toolkit for comparing differential gene expression profiles through gene set enrichment analysis. Starting from normalized microarray or RNA-seq gene expression values (stored in lists of ExpressionSet and CountDataSet objects) the package performs differential expression analysis using the limma or DESeq packages. Supplying a simple list of gene identifiers, global differential expression profiles or data from complete experiments as input, users can use a unified set of several well-known gene set enrichment analysis methods to retrieve experiments with similar changes in gene expression. To take into account the directionality of gene expression changes, gCMAPQuery introduces the SignedGeneSet class, directly extending GeneSet from the GSEABase package. To increase performance of large queries, multiple gene sets are stored as sparse incidence matrices within CMAPCollection eSets. gCMAP offers implementations of 1. Fisher's exact test (Fisher, J R Stat Soc, 1922) 2. The ''connectivity map'' method (Lamb et al, Science,2006) 3. Parametric and non-parametric t-statistic summaries (Jiang & Gentleman, Bioinformatics, 2007) and 4. Wilcoxon / Mann-Whitney rank sum statistics (Wilcoxon, Biometrics Bulletin, 1945) as well as

wrappers for the 5. camera (Wu & Smyth, Nucleic Acid Res, 2012) 6. mroast and romer (Wu et al, Bioinformatics, 2010) functions from the limma package and 7. wraps the gsea method from the mgsa package (Bauer et al, NAR, 2010). All methods return CMAPResult objects, an S4 class inheriting from AnnotatedDataFrame, containing enrichment statistics as well as annotation data and providing simple high-level summary plots.

License Artistic-2.0

LazyLoad yes

OS_type unix

ByteCompile TRUE

biocViews Bioinformatics, Microarray, Software, Pathways, Annotation

Collate 'AllClasses.R' 'AllGenerics.R' 'SignedGeneSet-accessors.R''utility-functions.R' 'camera_score-methods.R''connectivity_score-methods.R' 'featureScore-methods.R''fisher_score-methods.R' 'geneIndex-methods.R''gsealm_jg_score-methods.R' 'gsealm_score-methods.R''incidence-methods.R' 'mgsa_score-methods.R''mapIdentifiers-methods.R' 'minSetSize-methods.R''mroast_score-methods.R' 'romer_score-methods.R''wilcox_score-methods.R' 'CMAPCollection-accessors.R''CMAPResults-accessors.R'

R topics documented:

---

gCMAP-package *Tools for Connectivity Map-like analyses*

Description

The gCMAP package provides a toolkit for comparing differential gene expression profiles through gene set enrichment analysis. Starting from normalized microarray or RNA-seq gene expression values (stored in lists of ExpressionSet and CountDataSet objects) the package performs differential expression analysis using the limma or DESeq packages. Supplying a simple list of gene identifiers, global differential expression profiles or data from complete experiments as input, users can use a unified set of several well-known gene set enrichment analysis methods to retrieve experiments with similar changes in gene expression. To take into account the directionality of gene expression changes, gCMAPQuery introduces the SignedGeneSet class, directly extending GeneSet from the GSEABase package. To increase performance of large queries, multiple gene sets are stored as sparse incidence matrices within CMAPCollection eSets. gCMAP offers implementations of 1. Fisher's exact test (Fisher, J R Stat Soc, 1922) 2. The "connectivity map" method (Lamb et al, Science, 2006) 3. Parametric and non-parametric t-statistic summaries (Jiang & Gentleman, Bioinformatics, 2007) and 4. Wilcoxon / Mann-Whitney rank sum statistics (Wilcoxon, Biometrics Bulletin, 1945) as well as wrappers for the 5. camera (Wu & Smyth, Nucleic Acid Res, 2012) 6. mroast and romer (Wu et al, Bioinformatics, 2010) functions from the limma package and 7. wraps the gsea method from the mgsa package (Bauer et al, NAR, 2010). All methods return CMAPResult objects, an S4 class inheriting from AnnotatedDataFrame, containing enrichment statistics as well as annotation data and providing simple high-level summary plots.

Details

Package:	gCMAP
Type:	Package
Version:	1.3.5
Date:	2013-03-11
Depends:	GSEABase, limma (>= 3.15.14)

Imports:	Biobase, BiocGenerics, methods, GSEAlm, Category, bigmemory, bigmemoryExtras (>= 1.1.2), Matrix (>= 1
Suggests:	DESeq, KEGG.db, reactome.db, RUnit, GO.db, mgsa
License:	Artistic-2.0
LazyLoad:	yes
OS_type:	unix
ByteCompile:	TRUE
biocViews:	Bioinformatics, Microarray, Software, Pathways, Annotation
Collate:	'AllClasses.R' 'AllGenerics.R' 'SignedGeneSet-accessors.R' 'utility-functions.R' 'camera_score-methods.R
Built:	R 3.0.0; ; 2013-03-11 17:13:45 UTC; unix

Index:

gCMAP-package	Tools for Connectivity Map-like analyses
gCMAPData	Example 'NChannelSet'
geneIndex-methods	Methods for Function 'geneIndex' in Package 'gCMAP'
generate_gCMAP_NChannelSet	Generate a perturbation profile library from expression sets of control/treatment pairs
gsealm_jg_score-methods	Parametric test for testing normally distributed scores for gene set enrichment
gsealm_score-methods	Methods for Function 'gsealm_score' in Package 'gCMAP'
induceCMAPCollection-methods	Methods for Function 'induceCMAPCollection' in Package 'gCMAP'
mapNmerge	A function to map eSet featureNames and calculate summaries for many-to-one mapping features
matrix_or_big.matrix-class	Class '"matrix_or_big.matrix"'
memorize	Create a new NChannelSet instance by selecting specific channels and load BigMatrix assayData into memory, if present
mergeCMAPs	This function merged two eSets.
mgsa_score-methods	Model-based gene set analysis (MGSA)
minSetSize	GeneSetCollection length filtering
mroast_score-methods	Methods for Function 'mroast_score' in Package 'gCMAP'
pairwise_DESeq	Generate statistics associated with pairwise differential expression from RNAseq count data
pairwise_compare	Generate statistics associated with pairwise differential expression
romer_score-methods	Methods for Function 'romer_score' in Package 'gCMAP'
signedRankSumTest	An implementation of the Wilcox rank sum test / Mann-Whitney test that takes into account the direction / sign of gene set members and possibly the correlation between cases
splitPerturbations	Function to split an ExpressionSet downloaded from ArrayExpress based on the experimental factors present in the phenoData slot
wilcox_score-methods	Methods for Function 'wilcox_score' in Package 'gCMAP'
zScores	Function to calculate z-scores from p-values

Further information is available in the following vignettes:

diffExprAnalysis	main (source)
gCMAP	main (source)

Author(s)

Thomas Sandmann <sandmann.thomas@gene.com>, Richard Bourgon <bourgon.richard@gene.com> and Sarah Kummerfeld <kummerfeld.sarah@gene.com>

Maintainer: Thomas Sandmann <sandmann.thomas@gene.com>

`annotate_eset_list` *Function to complile a data frame with per-instance annotation for a list of eSet objects generated by the* `splitPerturbations` *function. The output can be used directly as sample.annotation for the* `NChannelSet` *function.*

Description

For each eSet in the 'eset.list', the pData slot is examined. Perturbation instances are identified as a match to 'perturbation' in the 'cmap.column' of the pData slot. The first matching row is extracted and transfered into the output data.frame, which contains one row for each eSet in the 'eset.list'. Only annotation columns found in the pData slots of all eSets in the 'eset.list' are returned.

Usage

```
annotate_eset_list(eset.list, cmap.column = "cmap", perturbation = "perturbation")
```

Arguments

`eset.list`	A list of eSet objects, usually generated by a call to the `splitPerturbations` functions
`cmap.column`	The name of the pData column for eSets in 'eset.list' identifying treatment and control samples.
`perturbation`	The character string in the 'cmap.column' of the pData column for eSets in 'eset.list' identifying perturbation associated with treated samples.

Value

A data frame with one row for each eSet in the input 'eset.list' and all columns found in the original eSet pData slot.

Author(s)

Thomas Sandmann

See Also

`splitPerturbations`

Examples

```
example( splitPerturbations )
```

---

camera_score-methods *Methods for Function* camera_score *in Package* **gCMAP**

Description

These methods provide a wrapper for the 'Competitive Gene Set Test Accounting for Inter-gene Correlation' function camera See 'limma' documention for details.

Usage

```
## S4 method for signature 'eSet,CMAPCollection'
camera_score(experiment,sets,predictor=NULL,
design.matrix=NULL, element="exprs",keep.scores=FALSE,...)

## S4 method for signature 'matrix,CMAPCollection'
camera_score(experiment, sets,...)

## S4 method for signature 'matrix,GeneSet'
camera_score(experiment,sets,...)

## S4 method for signature 'eSet,GeneSet'
camera_score(experiment, sets, element="exprs",...)

## S4 method for signature 'matrix,GeneSetCollection'
camera_score(experiment,sets,...)

## S4 method for signature 'eSet,GeneSetCollection'
camera_score(experiment, sets, element="exprs",...)
```

Arguments

sets	A CMAPCollection, GeneSetCollection or GeneSet object containing gene sets, with which to query the experiment object.
experiment	An eSet or data matrix with numeric data to compare the query object to.
predictor	A character vector or factor indicating the phenotypic class of the experiment data columns. Either the 'predictor' or 'design' parameter must be supplied.
design.matrix	A design matrix for the experiment. Either the 'predictor' or 'design' parameter must be supplied. If both are supplied, the 'design' is used.
element	Character vector specifying which channel of an eSet to extract (defaults to "exprs", alternatives may be e.g. "z", etc.)

keep.scores	Logical: keep gene-level scores for all gene sets (Default: FALSE) ? The size of the generated CMAPResults object increases with the number of contained gene sets. For very large collections, setting this parameter to 'TRUE' may require large amounts of memory.
...	Additional arguments passed to downstream methods.

Value

A CMAPResults object.

References

Wu, D, and Smyth, GK (2012). Camera: a competitive gene set test accounting for inter-gene correlation. Submitted.

Goeman, JJ, and Buhlmann, P (2007). Analyzing gene expression data in terms of gene sets: methodological issues. Bioinformatics 23, 980-987.

Examples

```
data(gCMAPData)
gene.set.collection <- induceCMAPCollection(gCMAPData, "z", higher=2, lower=-2)
sampleNames( gene.set.collection ) <- c("set1", "set2", "set3")

## random score matrix
y <- matrix(rnorm(1000*6),1000,6, dimnames=list(featureNames(gCMAPData), 1:6))

## set1 is differentially regulated
effect <- as.vector(members(gene.set.collection[,1]) * 2)
y[,4:6] <- y[,4:6] + effect

predictor <- c( rep("Control", 3), rep("Case", 3))

res <- camera_score(y, gene.set.collection, predictor = predictor, keep.scores=TRUE)
res

## heatmap of expression scores for set1
set1.expr <- geneScores(res)[["set1"]]
heatmap(set1.expr, scale="none", Colv=NA, labCol=predictor,
        RowSideColors=ifelse( attr(set1.expr, "sign") == "up", "red", "blue"),
        margin=c(7,5))
legend(0.35,0,legend=c("up", "down"), fill=c("red", "blue"), title="Annotated sign", horiz=TRUE, xpd=TRUE)
```

---

center_eSet *A function to to center columns of eSet channels on either their kernel density peak, their mean or their median.*

Description

This function works on the eSet assayDataElement specified as 'channel' and sweeps out either the 'peak', (max of the kernel density), 'mean' or 'median' statistic from each column. A modified eSet containing the centered assayDataElement is returned, with an additional .shift column included in the pData slot recording the shift statistic for each sample.

Usage

```
center_eSet(eset, channel, center = "peak", report.center=FALSE)
```

Arguments

eset	An eSet object
channel	A valid channel / AssayDataElementName of 'eset'
center	One of 'peak', 'mean', 'median' or 'none', specifying the statistic to sweep from each column of 'channel' in 'eset'. If 'peak', the max of the kernel density is determined and used a statistic in sweep. If 'none', the original 'eset' is returned.
report.center	Logical, include the shift applied to 'channel' in the pData slot of the returned NChannelSet ?

Value

An eSet of the same class as 'eset' with the centered 'channel' assayData slot. The swept-out statistic is recorded in the 'channel'.shift column of the phenoData slot. In addition, the median absolute deviation around the center is returned.

Author(s)

Thomas Sandmann

See Also

sweep

Examples

```
data( gCMAPData )

## column means of uncentered z-scores
round( apply( assayDataElement( gCMAPData, "z"), 2, mean, na.rm=TRUE), 2)

## column means of centered z-scores
centered <- center_eSet( gCMAPData, "z", "mean")
round( apply( assayDataElement( centered, "z"), 2, mean, na.rm=TRUE), 2)
```

CMAPCollection-class *Class* "CMAPCollection"

Description

An extension of the eSet class for the efficient storage of (large) gene set collections.

Objects from Class CMAPCollection

Objects can be created by calls of the form `new("CMAPCollection", assayData, phenoData, featureData,experimentD`
Alternatively, the user-friendly 'CMAPCollection' method is available.

The induceCMAPCollection function can be used to apply thresholds to numerical scores stored in eSet-like objects and returns a CMAPCollecion (see examples).

The CMAPCollection class is derived from the virtual eSet class. The assayData slot contains information about the membership of genes (rows) in gene sets (columns) in the form of an incidence matrix. The incidence matrix, accessible through the 'members' method, is a 'sparseMatrix' object, in which 1 / -1 entries identify gene set membership of up- and downregulated genes, respecively.

As opposed to the well-established GeneSetCollection class defined in the GSEABase package, the CMAPCollection class stores gene set membership in a matrix format, allowing direct access to individual gene sets as well as the relationships between different sets. The incidence matrix offers memory efficient storage of large gene set collection and can directly be used in matrix-based gene set analyses.

Through direct extension of the virtual eSet class, featureData and phenoData slots are available for storage of gene- and gene-set annotation, respectively. The column 'signed' in the phenoData slot indicates whether the different gene sets (columns) should be considered to be signed to disambiguate cases in which all gene set members are identified by a +1 entry. In this case, 'signed' = TRUE indicates that these genes should be considered upregulated members of the set (and no down-regulated members were identified / stored). If 'signed' = FALSE, no information about directionality is available, e.g. gene set members can be either up- or downregulated.

Slots

`assayData`: Object of class "AssayData"

`phenoData`: Object of class "AnnotatedDataFrame"

`featureData`: Object of class "AnnotatedDataFrame"

`experimentData`: Object of class "MIAxE"

`annotation`: Object of class "character"

`protocolData`: Object of class "AnnotatedDataFrame"

`.__classVersion__`: Object of class "Versions"

Extends

Class "eSet", directly. Class "VersionedBiobase", by class "eSet", distance 2. Class "Versioned", by class "eSet", distance 3.

Methods

geneIds `signature(object = "CMAPCollection")`: Returns a list of gene identifiers, with one list entry for each column of the assayDataSlot 'members'.

members `signature(object = "CMAPCollection")`: Returns the number of gene members in each gene set stored in the collection. For signed sets, also the number of up-/down-regulated members is returned.

members `signature(object = "CMAPCollection")`: Returns the coincidence matrix as stored in the assayData slot of the CMAPCollection as a sparseMatrix object (rows=genes, columns=gene sets).

signed `signature(object = "CMAPCollection")`: Returns the 'signed' column of the phenoData slot, indicating whether gene sets should be considered signed (TRUE) or un-signed (FALSE).

signed<- `signature(x = "CMAPCollection")`: Replacement method for the 'signed' column of phenoData.

minSetSize `signature(sets = "CMAPCollection")`: Filter CMAPcollection for minimum number of set members.

incidence `signature(x = "CMAPCollection")`: Returns in the transpose of the coincidence matrix stored in the assayData slot, mirroring the definition of 'incidence' for GeneSetCollections as defined in the GSEABase package.

mergeCollections `signature(x = "CMAPCollection", y = "CMAPCollection")`: Combines two CMAPCollections into one.

upIds `signature(x = "CMAPCollection")`: Returns the gene identifiers of all up-regulated gene set members (sign = 1).

downIds `signature(x = "CMAPCollection")`: Returns the gene identifiers of all down-regulated gene set members (sign = -1).

Note

The CMAPCollections class supports coercion from / to GeneSet and GeneSetCollection objects defined by the GSEABase package, as well as the SignedGeneSet derivative introduced by the gCMAP package itself.

Author(s)

Thomas Sandmann, sandmann.thomas@gene.com

See Also

`induceCMAPCollection`, `GeneSetCollection`, `SignedGeneSet`

Examples

```
## empty CMAPCollection
new("CMAPCollection")

## CMAPCollection from matrix
```

```
mat <- matrix( sample( c(-1,0,1), 100, replace=TRUE), ncol=10)
cmap <- CMAPCollection( mat )
members( cmap )

## CMAPCollection induced from NChannelSet
data( gCMAPData )
assayDataElementNames( gCMAPData )

cmap <- induceCMAPCollection(gCMAPData, "z", lower=-2, higher=2)
cmap
setSizes( cmap )
pData(cmap)
signed(cmap) <- c(TRUE, FALSE, TRUE)
signed(cmap)
head(members(cmap))

out <- fisher_score(cmap[,1], cmap, universe = featureNames( cmap))
out
```

CMAPResults-class *Class* "CMAPResults"

Description

This class serves as a container for the output of different gene-set enrichment analysis methods. It directly extends the AnnotatedDataFrame class by adding two additional slots ('docs' and 'errors' to store information about the analysis raun. Data for each queried gene set are stored in the 'data' slot of the AnnotatedDataFrame. Additional information about the data columns, e.g. the definition of 'effect' for the chosen analysis method, is available in the varMetadata slot of the AnnotatedDataFrame and can also be accessed through the 'labels' accessor function.

Details

The AnnotatedDataFrame 'table' is populated by gene set enrichment analysis methods as needed. Explicit accessor and replacement methods exist for the following columns:

- set: Identifiers of the tested gene sets (e.g. obtained from the sampleNames of an analyzed CMAPCollection object).
- trend: The direction of the detected effect, e.g. 'upregulated', 'overrepresented', etc.
- pval: The raw p-value of the observed effect. Default
- effect: The detected effect size, e.g. log odds ratio (returned by 'fisher_score) or summay t-statistic (returned by gsealm_jg_score), etc.
- nSet: The number genes in the query gene sets
- nFound: The number of query set genes detected in the target set, e.g. genes common to both sets. Default:NULL

In addition, gene set annotations can be included in further columns of the 'table' AnnotatedDataFrame, e.g. retrieved from the phenoData slot of a CMAPCollection.

Objects from the Class

Objects can be created by calls of the form `new("CMAPResults", ...)`. CMAPResults objects are usually created as output by gene set enrichment analysis methods.

Slots

`data`: A data.frame containing results for different gene sets (rows), with method-specific output stored in the columns.

`dimLabels`: A character vector of length 2 that provides labels for the rows and columns.

`varMetadata`: A data.frame with the number of rows equal to the number of columns in 'data' and at least one column, named 'labelDescription', containing additional information about each result column.

`.__classVersion__`: A 'Versions' object describing the R and Biobase version numbers used to created the instance. Intended for developer use.

`docs`: Object of class `"character"` Additional information about the analysis run, usually populated by the gene set enrichment method.

`errors`: Object of class `"list"` Intented for warnings or error messages associated with the results.

Methods

cmapTable `signature(object = "CMAPResults")`: Returns data and labels stored in the 'table' AnnotatedDataFrame. If no additional parameters are supplied, this method is synonymous with `pData(object@table)`.

Optional parameters: `n` (integer): the number of rows to return. `columns` (character): indicating which columns of the 'table' slot to include in the output.

docs `signature(object = "CMAPResults")`: Accessor method for the 'docs' slot.

docs<- `signature(x = "CMAPResults")`: Replacement method for the 'docs' slot.

effect `signature(object = "CMAPResults")`: Accessor method for the 'effect' column of the 'table' slot.

effect<- `signature(x = "CMAPResults")`: Replacement method for the 'effect' column of the 'table' slot.

errors `signature(object = "CMAPResults")`: Accessor method for the 'docs' slot.

errors<- `signature(x = "CMAPResults")`: Replacement method for the 'docs' slot.

labels `signature(object = "CMAPResults")`: Returns information about the data columns of the 'table' slot. Synonymous with `varMetadata(object@table)`.

labels<- `signature(x = "CMAPResults")`: Replacement method for the varMetdata slot of the 'table' AnnotatedDataFrame. Replacement value must be a data.frame with as many rows as there are columns in 'table' and contain the column 'labelDescription'. See `AnnotatedDataFrame` for details.

nFound `signature(object = "CMAPResults")`: Accessor method for the 'nFound' column of the 'table' slot.

nFound<- `signature(x = "CMAPResults")`: Replacement method for the 'nFound' column of the 'table' slot.

nSet `signature(object = "CMAPResults")`: Accessor method for the 'nSet' column of the 'table' slot.

nSet<- `signature(x = "CMAPResults")`: Replacement method for the 'nSet' column of the 'table' slot.

padj `signature(object = "CMAPResults")`: Accessor method for the 'padj' column of the 'table' slot.

padj<- `signature(x = "CMAPResults")`: Replacement method for the 'padj' column of the 'table' slot.

plot `signature(x = "CMAPResults", y = "ANY")`: Returns an overview of the results stored in a CMAPResults object: 1. the distribution of scores across all results and 2. a heatmap of rank-ordered effect sizes.

- strip.effect: String specifying the CMAPResults column to retrieve scores from. Default:"effect"
- strip.pval: String specifying the CMAPResults column to transform into unsigned z-scores. Only evaluated if 'density.effect' column is not present or is set to 'NULL'.Default:"padj"
- strip.cutoffs: Numeric vector of length 2. Scores between strip.cutoffs[1] and strip.cutoffs[2] will be displayed in strip.col[2]. Default:c(-3,3)
- strip.bounds: Numeric vector of length 2 specifying the end points of the color gradient. Scores < strip.cutoffs[1] or > strip.cutoffs[2] will not be distinguishable.Default:c(-6,6)
- strip.col: Vector of length 3, specifying the colors used in the heatmap strip: strip.col[1] = low scores, strip.col[2] = medium score (excluded from gradient), strip.col[3] = high scores. Default:c("blue","white","red")
- set.inf: Numerical replacing Inf/-Inf scores in the density plot (default:+/-20)
- col.upRug plot color for positively correlated instances
- col.downRug plot color for negatively correlated instances

pval `signature(object = "CMAPResults")`: Accessor method for the 'pval' column of the 'table' slot.

pval<- `signature(x = "CMAPResults")`: Replacemant method for the 'pval' column of the 'table' slot

set `signature(object = "CMAPResults")`: Accessor method for the 'set' column of the 'table' slot.

set<- `signature(x = "CMAPResults")`: Replacement method for the 'set' column of the 'table' slot.

show `signature(object = "CMAPResults")`: Returns a summary of the CMAPResult object, including the number rows in the 'table' slot and shows the top five results with the smallest p-values.

trend `signature(object = "CMAPResults")`: Accessor method for the trend' column of the 'table' slot.

trend<- `signature(x = "CMAPResults")`: Replacement method for the geneScores' column of the 'table' slot.

geneScores `signature(object = "CMAPResults")`: Accessor method for the geneScores' column of the 'table' slot. When available, this column stores a list of matrices, one for each row of the CMAPResults object, with raw per-gene scores for all members of the gene set. While the 'show' method displays only a brief summary of the available data, the geneScores method retrieves the full list of score matrices.

zscores `signature(x = "CMAPResults")`: Transforms adjusted p-values stored in a CMAPResults into z-scores based on the standard normal distribution.

Author(s)

Thomas Sandmann, sandmann.thomas@gene.com

See Also

AnnotatedDataFrame

Examples

```
## create random score profile
set.seed(123)
z <- rnorm(1000)
names(z) <- paste("g", 1:1000, sep="")

## generate random incidence matrix of gene sets
m <-replicate(1000, {
  s <- rep(0,1000)
  s[ sample(1:1000, 20)] <-  1
  s[ sample(1:1000, 20)] <- -1
  s
  })
dimnames(m) <- list(names(z), paste("set", 1:1000, sep=""))

## Set1 is up-regulated
z <- z + m[,1]*2

## create CMAPCollection
cmap <- CMAPCollection(m, signed=rep(TRUE,1000))

## gene-set enrichment test
res <- gsealm_jg_score(z, cmap)
class(res)
res

## overview plot
plot(res)

## rerun, this time store gene-level scores
res <- gsealm_jg_score(z, cmap, keep.scores=TRUE)
res
m <- geneScores(res)
m[["set1"]] ## scores for set1

## stripplot for set1, colored by annotated sign
gene.signs <- factor( attr(m[["set1"]], "sign"))
boxplot( m[["set1"]] ~ gene.signs,
  ylab="z-score",
  main="Set1",
  col=c("blue", "red"))
```

connectivity_score *Broad CMAP gene set enrichment metrics*

Description

A method for computing Broad CMAP connectivity scores, as described in the reference below. Supporting functions used for computation are also described.

Usage

```
## S4 method for signature 'eSet,CMAPCollection'
connectivity_score(experiment, query, element="z", keep.scores=FALSE)

## S4 method for signature 'matrix,CMAPCollection'
connectivity_score(experiment, query, ...)

## S4 method for signature 'eSet,SignedGeneSet'
connectivity_score(experiment, query, ...)

## S4 method for signature 'matrix,SignedGeneSet'
connectivity_score(experiment, query, ...)

## S4 method for signature 'eSet,GeneSetCollection'
connectivity_score(experiment, query,
...)

## S4 method for signature 'matrix,GeneSetCollection'
connectivity_score(experiment, query,...)

## S4 method for signature 'ANY,GeneSet'
connectivity_score(experiment, query, ...)
```

Arguments

experiment	An eSet or matrix object to query.
query	A CMAPCollection, SignedGeneSet, or GeneSetCollection object containing signed gene sets with which to query the experiment object.
element	Character string specifying which element of a multi-channel eSet to access for determining tag rank?
keep.scores	Logical: keep gene-level scores for all gene sets (Default: TRUE) ? The size of the generated CMAPResults object increases with the number of contained gene sets. For very large collections, consider setting this parameter to 'FALSE' to conserve memory.
...	Additional arguments passed on to downstream functions.

Value

connectivity_score
: For the SignedGeneSet method, a vector of s scores, one per instance in experiment. For the GeneSetCollection method, a matrix, with one row per instance in experiment and one column per query set.

ks
: A signed Kolmogorov-Smirnov type statistic based on the position of the ranks V in the vector 1:n.

s
: A difference of ks values for V_up vs. V_down, or 0 if both yield the same sign.

S
: A vector of signed, rescaled scores. After rescaling, 1 corresponds to the maximum positive s score, and -1, to the minimum negative s score. S is typically used to produce the red-grey-green instance heat maps from the reference below.

Note

Note that as defined by Lamb et al., ks is not symmetric. For n = 100, for example, ks(1,100) is .99 while ks(100,100 is -1. A further consequence of the Lamb et al. definitions is that the maximum possible score for a perfect positive match depends on query set size. See the example below. Although these properties are not desirable, the intention here is to exactly reproduce the Lamb et al. statistic.

References

Lamb, J. et al. (2006). The Connectivity Map: using gene-expression signatures to connect small molecules, genes, and disease. *Science* 313:1929. Notation for ks, s, and S closely follows the Supporting Online Material there.

Examples

```
data(gCMAPData)

  ## induce CMAPCollection from z-scores
  sets <- induceCMAPCollection(gCMAPData, "z", lower=-3, higher=3)

  ## Broad CMAP KS scoring: one z-score column
  connectivity_score(gCMAPData[,1], sets, element="z")

  ## multiple z-score columns, results are returned as a list
  connectivity_score(gCMAPData, sets)
```

DESeq_nbinom *Function to perform a DESeq analysis to detect differential expression between perturbation and control groups.*

Description

This function is a wrapper for a standard DESeq analyis with two classes, perturbation and control, annotated in the 'conditions' column of the cds phenoData slot. First , the size factors are determined using default parameters. Next, a dispersion parameter is estimated using the default (pooled) method. Finally, p-values are estimated for differential expression between treatment and control groups.

Usage

```
.DESeq_nbinom(cds, control = "control", perturb = "perturbation",
try.hard = FALSE, control_perturb_col = "cmap",...)
```

Arguments

`cds`	A CountDataSet with perturbation and control samples identified in the pData condition slot.
`control`	Character string corresponding to the control factor level of the condition phenoData slot.
`perturb`	Character string corresponding to the perturbation factor level of the condition phenoData slot.
`try.hard`	Logical parameter indicating the function's behavior in case the parametric (default) dispersion estimation fails. If FALSE (default), the function exits with an error. If TRUE, a non-parametric (loess) esimation is attempted instead.
`control_perturb_col`	Column name in `phenoData` of `cds` where control/perturbation designations are stored.
`...`	Any additional parameters passed on to estimateDispersions

Value

See nbinomTest for details.

Note

To use this function, please install the suggested Bioconductor package 'DESeq'.

`eSetOnDisk` *A function to store the assayData of an eSet object as BigMatrix files on disk.*

Description

This function accepts and eSet object and generates separate file-backed BigMatrix objects for each assayDataElement. Data is only loaded into memory upon subsetting, allowing the retrieval of selected data without loading the (potentially large) object into memory in full.

Usage

```
eSetOnDisk(eset, out.file = NULL)
```

Arguments

eset	An object inheriting from eSet.
out.file	The path and basename of the output file. Three files will be generated for each eSet assayDataElement, identified by extending 'out.file' by suffices.

Value

An object of the same class as 'eset', with BigMatrix elements in the assayData slot.

Note

Please see the BigMatrix package for more details on BigMatrix objects.

Author(s)

Thomas Sandmann

See Also

BigMatrix eSet memorize

Examples

```
## load ExpressionSet
data("sample.ExpressionSet") ## from Biobase
sample.ExpressionSet ## two assayDataElements: exprs, se

## 'exprs' data matrix
class( assayDataElement( sample.ExpressionSet, "exprs" ) )

## convert assayData to BigMatrix objects
storage.file <- tempfile() ## create path & basemane for BigMatrices
storage.file

on.disk <- eSetOnDisk( sample.ExpressionSet, storage.file )
on.disk ## ExpressionSet
dir(dirname( storage.file )) ## created 3 files per channel

class( assayDataElement( on.disk, "exprs" ) ) ## BigMatrix object

## BigMatrix objects are loaded only upon subsetting
assayDataElement( on.disk, "exprs") ## retrieves BigMatrix, NOT matrix
assayDataElement( on.disk, "exprs")[1:10,1:10] ## loads subset only
dim( assayDataElement( on.disk, "exprs")[,] ) ## retrieves full matrix

## convert back to standard in-memory ExpressionSet
```

```
in.memory <- memorize( on.disk )
class( assayDataElement( in.memory, "exprs" ) ) ## standard matrix object

## remove tempfiles generated for this example
unlink(paste(storage.file,"*", sep=""))
```

`eset_instances` *A function to subset an eSet with expression data into smaller datasets, each corresponding to a single perturbation experiment.*

Description

This function takes two parameters, an eSet object (e.g. an ExpressionSet or CountDataSet) containing muliple samples, and a numerical matrix defining how these samples should be compared to investigate perturbations of interest. For each perturbation, a separate eSet object is generated, ready for analysis with the `generate_gCMAP_NChannelSet` function. Samples can be used in multiple instances, e.g. common controls can be specified in each column of the 'instance.matrix'.

Usage

```
eset_instances(instance.matrix, eset, control_perturb_col = "cmap", control = "control", perturb = "per
```

Arguments

`instance.matrix`	A numeric matrix of -1 and 1's. Each columns defines a contrast of interest and indicates whether a sample (row) corresponds to a control sample (-1) or a perturbation sample (1). The row.names of the instance.matrix correspond to sampleNames of 'eset'. Entries other than -1 or 1 will be ignored.
`eset`	An eSet object to be subset into smaller datasets. The row.names of 'eset' must correspond to the row.names of the 'instance.matrix'.
`control_perturb_col`	Character, indicating which phenoData column to use to store 'control' and 'perturb' labels.
`control`	Character, defining the label stored in each new eSet to indicate control samples.
`perturb`	Character, defining the label stored in each new eSet to indicate perturbation samples.

Value

A list of eSet objects, each corresponding to one instance defined by the columns of 'incidence.matrix'.

Note

This function can be used to generate the 'eset.list' required for differential expression analyses with `generate_gCMAP_NChannelSet`.

Author(s)

Thomas Sandmann, sandmann.thomas@gene.com

See Also

generate_gCMAP_NChannelSet

Examples

```
library(Biobase)
data(sample.ExpressionSet)

## contains Male/Female and Control/Case annotations
pData( sample.ExpressionSet)

## separate analysis of Male/Female patients
male <- ifelse( pData( sample.ExpressionSet )$type == "Control", -1, 1)
male[which( pData( sample.ExpressionSet )$sex == "Female")] <- 0

female <- ifelse( pData( sample.ExpressionSet )$type == "Control", -1, 1)
female[which( pData( sample.ExpressionSet )$sex == "Male")]  <- 0

instance.matrix <- cbind( male, female)
row.names( instance.matrix ) <- sampleNames( sample.ExpressionSet )

eset_instances( instance.matrix, sample.ExpressionSet)
```

---

featureScores-methods *Methods to obtain scores for CMAPCollection gene sets from a matrix or eSet*

Description

These methods extract the scores for CMAPCollection gene set members from eSet or matrix objects and return them as a list (argument 'query') of lists (argument 'dat') with score vectors. Argument order determines the organization of the list, e.g. if 'query' is a CMAPCollection, one list element is returned for each gene set, containing all score vectors for the respective set. If 'simplify' is set to TRUE, score vectors are combined and a list of matrices is returned instead. Score vectors and matrices carries an additional 'sign' attribute corresponding to the sign annotated in the CMAPCollection.

Usage

```
## S4 method for signature 'CMAPCollection,eSet'
featureScores(query, dat, element="z",simplify=TRUE)

## S4 method for signature 'CMAPCollection,matrix'
```

```
featureScores(query,dat, simplify=TRUE)

## S4 method for signature 'CMAPCollection,BigMatrix'
featureScores(query, dat, simplify=TRUE)

## S4 method for signature 'eSet,CMAPCollection'
featureScores(query, dat, element="z")

## S4 method for signature 'matrix,CMAPCollection'
featureScores(query, dat)

## S4 method for signature 'BigMatrix,CMAPCollection'
featureScores(query, dat)

## S4 method for signature 'CMAPCollection,numeric'
featureScores(query, dat)

## S4 method for signature 'CMAPCollection,CMAPCollection'
featureScores(query, dat)

## S4 method for signature 'numeric,CMAPCollection'
featureScores(query, dat)

## S4 method for signature 'CMAPCollection,matrix_or_big.matrix'
featureScores(query, dat, simplify = TRUE)

## S4 method for signature 'matrix_or_big.matrix,CMAPCollection'
featureScores(query, dat)
```

Arguments

query	A CMAPCollection, eSet or matrix.
dat	A CMAPCollection, eSet or matrix.
element	Character string specifying which assayDataElement to extract from eSet objects.
simplify	Logical: when possible, should score columns for each gene set collected in a matrix ?

Value

A nested list: one list element for each 'query', containing a list with score vectors for each 'dat'.

Methods

```
signature(query = "CMAPCollection", dat = "eSet") }
signature(query = "CMAPCollection", dat = "matrix") }
signature(query = "CMAPCollection", dat = "BigMatrix") }
```

```
signature(query = "eSet", dat = "CMAPCollection") }
signature(query = "matrix", dat = "CMAPCollection") }
signature(query = "BigMatrix", dat = "CMAPCollection") }
signature(query = "CMAPCollection", dat = "numeric") }
signature(query = "CMAPCollection", dat = "CMAPCollection") }
signature(query = "numeric", dat = "CMAPCollection") }
signature(query = "matrix_or_big.matrix", dat = "CMAPCollection") }
signature(query = "CMAPCollection", dat = "matrix_or_big.matrix") }
```

Author(s)

Thomas Sandmann

Examples

```
data(gCMAPData)
## generate CMAPCollection with two sets (drug1, drug2)
sets <- induceCMAPCollection(gCMAPData, "z", higher=-2, lower=2)[,1:2]
sampleNames(sets) <- c("set1", "set2")

## extract per-gene scores as matrices
res <- featureScores(sets, gCMAPData)
class(res) ## list
names(res) ## one element per set
class(res[["set1"]]) ## matrix
dim(res[["set1"]])

## or as lists of score vectors
res2 <- featureScores(sets, gCMAPData, simplify=FALSE)
class(res2[["set1"]]) ## list
length(res2[["set1"]])

## stripplot for set2, colored by annotated sign
m <- res[["set2"]][,"drug2"]
colors <- ifelse( attr(res[["set2"]], "sign") == "up", "red", "blue")
gene.sign <- factor( attr(res[["set2"]], "sign"))
boxplot(
  m ~ gene.sign,
  col=c("blue", "red"),
  ylab="z-score",
  xlab="Query gene sign",
  main="Set 2"
  )
```

`fisher_score-methods` *Hypergeometric probability of gene set enrichment*

Description

A method for computing enrichment probilities based on the hypergeometric distribution. This method performs an over-representation analysis by generating 2x2 incidence matrices for gene sets provided as 'query' and 'sets' as GeneSet, SignedGeneSet, GeneSetCollection or CMAPCollection objects. If 'sets' is an NChannelSet object with quantitative data, gene sets are induced on the fly from the channel specified by the 'element' parameter.

Arguments

`query`	A `CMAPCollection`, `GeneSet`, or `GeneSetCollection` object containing the 'query' gene sets to compare against the 'sets'
`sets`	A `CMAPCollection`, `GeneSetCollection` or `GeneSet` object
`universe`	A character string of gene ids for all genes that could potentially be of interest, e.g. all genes represented on a microarray, all annotated genes, etc.
`keep.scores`	Logical: store the identifiers for the genes detected in 'query' and 'sets' ? (Default: FALSE) The size of the generated CMAPResults object increases with the number of contained gene sets. For very large collections, setting this parameter to 'TRUE' may require large amounts of memory.
`element`	A character string corresponding to the assayDataElementName of the NChannelSet object to be thresholded on the fly with the `induceCMAPCollection`.
`lower`	The lower threshold for the `induceCMAPCollection`.
`higher`	The 'higher' threshold for the `induceCMAPCollection`.
`min.set.size`	Number of genes a gene set induced by `induceCMAPCollection` needs to contain to be included in the analysis (Default:5).
`...`	Additional arguments passed to downstream methods.

Value

A CMAPResults object

Methods

```
signature(query = "CMAPCollection", sets = "CMAPCollection", universe = "character")

signature(query = "CMAPCollection", sets = "NChannelSet", universe = "character")

signature(query = "SignedGeneSet", sets = "CMAPCollection", universe = "character")

signature(query = "SignedGeneSet", sets = "NChannelSet", universe = "character")
```

```
signature(query = "GeneSet", sets = "CMAPCollection", universe = "character")
signature(query = "GeneSet", sets = "NChannelSet", universe = "character")
signature(query = "GeneSetCollection", sets = "CMAPCollection", universe = "character")

signature(query = "GeneSetCollection", sets = "NChannelSet", universe = "character")

signature(query = "GeneSet", sets = "GeneSetCollection", universe = "character")

signature(query = "CMAPCollection", sets = "GeneSetCollection", universe = "character")

signature(query = "GeneSetCollection", sets = "GeneSetCollection", universe = "character")

signature(query = "GeneSet", sets = "GeneSet", universe = "character")
```

Note

p-values are corrected for multiple testing separately for each query set, but not across multiple queries.

See Also

fisher.test

Examples

```
data(gCMAPData)

gene.set.collection <- induceCMAPCollection(gCMAPData, "z", higher=2, lower=-2)

## compare all gene sets in the gene.set.collection to each other
universe = featureNames(gCMAPData)
fisher_score(gene.set.collection, gene.set.collection, universe = universe)
```

---

gCMAPData *Example* NChannelSet

Description

The gCMAPData object is an NChannelSet object with 3 samples x 1000 features x 3 channels (p-value, z-score and log_fc).

Usage

```
data(gCMAPData)
```

Examples

```
data(gCMAPData)

gene.set.collection <- induceCMAPCollection(gCMAPData, "z", higher=2, lower=-2)

## comparison with a single user-provided profile of z-scores
profile <- assayDataElement(gCMAPData, "z")[,1]
gsealm_jg_score(profile, gene.set.collection)
```

---

geneIndex-methods *Methods for Function* geneIndex *in Package* **gCMAP**

Description

These methods match a character vector of gene ids to the members of a GeneSet, GeneSetCollection or CMAPCollection and return the match indices.

Usage

```
## S4 method for signature 'CMAPCollection,character'
geneIndex(gene.sets,gene.ids,remove.empty=TRUE)

## S4 method for signature 'GeneSetCollection,character'
geneIndex(gene.sets,gene.ids,remove.empty=TRUE)

## S4 method for signature 'GeneSet,character'
geneIndex(gene.sets,gene.ids,remove.empty=TRUE)
```

Arguments

gene.sets	A CMAPCollection, GeneSetCollection or GeneSet to match the 'gene.ids' against.
gene.ids	A character string of gene identifiers whose position (if any) in the 'gene.sets' is to be determined.
remove.empty	Logical parameter specifying whether gene sets without any matching gene.ids should be removed from the output.

Value

An integer vector or (if a collection was searched) a list of integer vectors with the matching positions of gene.ids in the gene.sets.

Examples

```
## induce CMAPCollection
data(gCMAPData)
gene.set.collection <- induceCMAPCollection(gCMAPData, "z", higher=2, lower=-2)

gene.ids <- geneIds(gene.set.collection[,2]) ## geneIds of the second set
geneIndex(gene.set.collection, gene.ids)
```

`generate_gCMAP_NChannelSet` *Generate a perturbation profile library from expression sets of control/treatment pairs*

Description

When provided with a list of ExpressionSet or countDataSet objects, comparisons are made between control and perturbation samples on a set basis. To process RNAseq count data, the suggested Bioconductor package 'DESeq' must be available on the system. For countDataSets, a moderated log2 fold change for each set is calculated after variance-stabilizing transformation of the count data is performed globally across all countDataSets in the list.

Usage

```
generate_gCMAP_NChannelSet(
                           data.list,
                           uids=1:length(data.list),
                           sample.annotation=NULL,
                           platform.annotation="",
                           control_perturb_col="cmap",
                           control="control",
                           perturb="perturbation",
                           limma=TRUE,
                           limma.index=2,
                           big.matrix=NULL,
                           center.z="peak",
                           center.log_fc="none",
                           report.center=FALSE
                           )
```

Arguments

`data.list`	List of `ExpressionSet` or CountDataSet objects. Each element includes all array / RNAseq data for a single instance, plus metadata on which samples are perturbation and control.
`uids`	Vector of unique identifiers for the instances in `data.list`

`sample.annotation`
: An optional `data.frame` of additional annotation for instances, each row corresponds to one instance, ordered to correspond with the `data.list`. This is not used for the control/perturbation comparisons, instead it is simply attached to the `NChannelSet` for future reference.

`platform.annotation`
: The name of the platform as used by the annotation package.

`control_perturb_col`
: See `pairwise_compare`.

`control`
: See `pairwise_compare`.

`perturb`
: See `pairwise_compare`.

`limma`
: Use limma package to perform moderated t-tests (Default: TRUE) instead of a standard t-test ?

`limma.index`
: Integer specifying the index of the parameter estimate for which we to extract t and other statistics. The default corresponds to a two-class comparison with the standard parameterization. The function assumes that there was no missing data, so that test for all genes were performed on the same sample size.

`big.matrix`
: Character string providing the path and filename to store the NChannelSets on disk instead of in memory. If 'NULL' (default), an NChannelSet is returned. If not 'NULL', the BigMatrix package will create (or overwrite !) three binary files for each channel of the NChannelSet at the location provided as 'big.matrix', distinguishing files for the different channes by their suffices. To load the NChannelSet into a different R session, the binary files must be accessible.

`center.z`
: One of 'none', 'peak', 'mean', 'median', selecting whether / how to center the z-scores for each experiment. Option 'peak' (default) will center on the peak of the z-score kernel density. Options 'mean' and 'median' will center on their respective values instead.

`center.log_fc`
: One of 'none', 'peak', 'mean', 'median', selecting whether / how to center the log2 fold-change distribution for each experiment. Option 'peak' will center on the peak of the z-score kernel density. Options 'mean' and 'median' will center on their respective values instead.

`report.center`
: Logical, include the z-score / log2 fold change corrections and the median absolute deviation of the respective distribution about zero in the pData slot of the returned NChannelSet ?

Value

The function returns an `NChannelSet` with one channel for each of the columns returned by `pairwise_compare`. This can be worked with directly (e.g, `assayData(obj)$z`), or specific channels can be converted to regular `ExpressionSet` objects (e.g.,`es <- channel(obj, "z")`). In the latter case, one would access `z` by `exprs(es)`. If 'report.center' is TRUE, the pData slot of the NChannelSet contains columns reporting the shift applied to the z-score and / or log2 fold change columns to center the score distributions on zero and the median absolute deviation of the shifted distribution about zero.

Examples

```
## list of ExpressionSets
data("sample.ExpressionSet") ## from Biobase

es.list <- list( sample.ExpressionSet[,1:4], sample.ExpressionSet[,5:8], sample.ExpressionSet[,9:12])
names(es.list) <- paste( "Instance", 1:3, sep=".")

de <- generate_gCMAP_NChannelSet(
    es.list,
    1:3,
    platform.annotation = annotation(es.list[[1]]),
    control_perturb_col="type",
    control="Control",
    perturb="Case")

assayDataElementNames(de)
head( assayDataElement(de, "z") )

## Not run:
## processing RNAseq data requires the suggested 'DESeq'
## Bioconductor package.
require( DESeq )
set.seed( 123 )
## list of CountDataSets
cds.list <- lapply( 1:3, function(n) {
   cds <- makeExampleCountDataSet()
   featureNames(cds) <- paste("gene",1:10000, sep="_")
   cds
})

cde <- generate_gCMAP_NChannelSet(cds.list,
                           uids=1:3,
                           sample.annotation=NULL,
                           platform.annotation="Entrez",
                           control_perturb_col="condition",
                           control="A",
                           perturb="B")

assayDataElementNames(cde)

## End(Not run)
```

GeneSet *Methods for* GeneSet *and* GeneColorSet

Description

Additional methods for function GeneSet and GeneColorSet, supporting use of NChannelSet templates.

Note

The ExpressionSet methods are used verbatim, since only metadata is utilized. Note that collectionType will be ExpressionSet as a result.

---

gsealm_jg_score-methods

Parametric test for testing normally distributed scores for gene set enrichment

---

Description

This method implements the 'JG' summary method introduced by Oron et al, 2008, as a stand-alone method to query a set of normally-distributed scores (e.g. t-statistics or z-scores) with gene sets (and vice versa).

Scores for gene-set members are summed, respecting their sign (up- or down-regulated) and the combined score is divided by the square-root of the number of set members.

To fit linear models to an expression profiling experiment instead, please use the gsealm_jg_score method instead.

Arguments

query	An eSet, CMAPCollection, GeneSetCollection,GeneSet, matrix or numeric vector with data and gene ids. If a matrix is provided, gene ids must be provided as row-names. If a vector is provided, gene ids must be provided as names.
sets	See 'query'
removeShift	Optional parameter indicating that the aggregated test statistic should be centered by subtracting the column means (default=FALSE).Note: this option is not available for analysis of big.matrix backed eSet objects.
element	For eSet objects, which assayDataElement should be extracted ? (Default="exprs")
keep.scores	Logical: keep gene-level scores for all gene sets (Default: FALSE) ? The size of the generated CMAPResults object increases with the number of contained gene sets. For very large collections, setting this parameter to 'TRUE' may require large amounts of memory.
...	Additional arguments to be passed on to downstream methods.

Value

A CMAPResults object or, in case of multi-dimensional queries, a list of CMAPResults objects.

Methods

```
signature(query = "CMAPCollection", sets = "eSet")
signature(query = "CMAPCollection", sets = "matrix")
signature(query = "CMAPCollection", sets = "numeric")
signature(query = "eSet", sets = "CMAPCollection")
signature(query = "eSet", sets = "GeneSet")
signature(query = "eSet", sets = "GeneSetCollection")
signature(query = "GeneSet", sets = "eSet")
signature(query = "GeneSet", sets = "matrix")
signature(query = "GeneSet", sets = "numeric")
signature(query = "GeneSetCollection", sets = "eSet")
signature(query = "GeneSetCollection", sets = "matrix")
signature(query = "GeneSetCollection", sets = "numeric")
signature(query = "matrix_or_big.matrix", sets = "CMAPCollection")
signature(query = "matrix", sets = "GeneSet")
signature(query = "matrix", sets = "GeneSetCollection")
signature(query = "numeric", sets = "CMAPCollection")
signature(query = "numeric", sets = "GeneSet")
signature(query = "numeric", sets = "GeneSetCollection")
```

References

Gene set enrichment analysis using linear models and diagnostics. Oron AP, Jiang Z, Gentleman R. Bioinformatics. 2008 Nov 15;24(22):2586-91. Epub 2008 Sep 11.

Extensions to gene set enrichment. Jiang Z, Gentleman R. Bioinformatics. 2007 Feb 1;23(3):306-13. Epub 2006 Nov 24.

Examples

```
data(gCMAPData)
gene.set.collection <- induceCMAPCollection(gCMAPData, "z", higher=2, lower=-2)

## comparison with a single user-provided profile of z-scores
profile <- assayDataElement(gCMAPData, "z")[,1]
gsealm_jg_score(profile, gene.set.collection)

## comparison with of multiple profiles of z-scores to the CMAPCollection
res <- gsealm_jg_score(assayDataElement(gCMAPData, "z"), gene.set.collection)

## first CMAPResult object
res[[1]]

## adjusted p-values from all CMAPResult objects
sapply(res, padj)
```

```
## inverted query: CMAPCollection is compared to z-score profiles
gsealm_jg_score(gene.set.collection, assayDataElement(gCMAPData, "z"))[[1]]
```

---

gsealm_score-methods *Methods for Function* gsealm_score *in Package* **gCMAP**

Description

This method extends functions from the GSEAlm package to perform label-permutation based differential expression analysis. In addition to gene set membership, information about the gene sign (up- or down-regulated) is taken into consideration.

Usage

```
## S4 method for signature 'ExpressionSet,CMAPCollection'
gsealm_score(query, set,
removeShift=FALSE, predictor=NULL, formula=NULL, nPerm=1000, parametric=FALSE,respect.sign=TRUE, keep

## S4 method for signature 'eSet,CMAPCollection'
gsealm_score(query, set, element="exprs", ... )

## S4 method for signature 'matrix,CMAPCollection'
gsealm_score(query, set, predictor=NULL, ...)

## S4 method for signature 'eSet,GeneSetCollection'
gsealm_score(query, set, element="exprs",...)

## S4 method for signature 'matrix,GeneSetCollection'
gsealm_score(query, set, ...)

## S4 method for signature 'ExpressionSet,GeneSet'
gsealm_score(query, set,...)

## S4 method for signature 'ExpressionSet,GeneSetCollection'
gsealm_score(query, set,...)

## S4 method for signature 'eSet,GeneSet'
gsealm_score(query, set, element="exprs", ...)

## S4 method for signature 'matrix,GeneSet'
gsealm_score(query, set, ...)
```

Arguments

query	An ExpressionSet or matrix with normalized expression data.

set	A CMAPCollection, GeneSetCollection or GeneSet object containing gene sets. Gene ids must match those of the 'query'
removeShift	logical: should normalization begin with a column-wise removal of the mean shift? Note: this option is not available for analysis of big.matrix backed eSet objects.
predictor	A character string identifying one column in the pData slot of a 'query' ExpressionSet from which to construct the formula for the linear model. Ignored if 'formula' is provided.
formula	The formula to be used in the linear model. See gsealmPerm for details.
nPerm	The number of sample-label permutations to perform.
parametric	Logical, if set to 'TRUE', no label permutations are performed. Instead, p-values are calculated based on a parametric approximation.
respect.sign	Logical, if set to 'FALSE', gene sign information is ignored, considering up- and down-regulated genes to be equal.
element	Character string specifying which element to extract when coercing an ExpressionSet from an eSet subclass.
keep.scores	Logical: keep gene-level scores for all gene sets (Default: FALSE) ? The size of the generated CMAPResults object increases with the number of contained gene sets. For very large collections, setting this parameter to 'TRUE' may require large amounts of memory.
...	Additional arguments passed on to downstream functions.

Value

This method returns a CMAPResults object.

See Also

gsealmPerm lmPerGene

Examples

```
data(gCMAPData)

## induce gene sets from a collection of z-scores
gene.set.collection <- induceCMAPCollection(gCMAPData, "z", higher=2, lower=-2)
sampleNames(gene.set.collection) <- c("set1", "set2", "set3")

## random score matrix
y <- matrix(rnorm(1000*6),1000,6, dimnames=list(featureNames(gCMAPData), 1:6))

## set1 is differentially regulated
effect <- as.vector(members(gene.set.collection[,1]) * 2)
y[,4:6] <- y[,4:6] + effect

predictor <- c( rep("Control", 3), rep("Case", 3))

## run analysis and keep gene-level expression scores
```

```
res <- gsealm_score(y, gene.set.collection, predictor=predictor, nPerm=100, keep.scores=TRUE)
res

## heatmap of expression scores for set1
set1.expr <- geneScores(res)[["set1"]]
heatmap(set1.expr, scale="none", Colv=NA, labCol=predictor,
        RowSideColors=ifelse( attr(set1.expr, "sign") == "up", "red", "blue"),
        margin=c(7,5))
legend(0.35,0,legend=c("up", "down"), fill=c("red", "blue"), title="Annotated sign", horiz=TRUE, xpd=TRUE)
```

`induceCMAPCollection-methods`

Methods for Function `induceCMAPCollection` *in Package* **gCMAP**

Description

This method defines a `CMAPCollection` by applying thresholds to an element of an eSet-derived object. For example, applying 'induceCMAPCollection' to a matrix of z-scores stored in an NChannelSet, gene sets can be defined for each of the sample columns stored in the object. (See example section).

Usage

```
## S4 method for signature 'eSet'
induceCMAPCollection(eset,element,lower=NULL,higher=NULL,sign.sets=TRUE)
## S4 method for signature 'matrix'
induceCMAPCollection(eset,element,...)
```

Arguments

`eset`	An object derived from class `eSet`, e.g. an NChannelSet
`element`	A character string corresponding to the assayDataElementName of the 'eset' object to which the thresholds should be applied.
`lower`	The lower threshold. If not 'NULL', genes with a score smaller than 'lower' will be included in the gene set with sign -1. At least one of 'lower' and 'higher' must be specified.
`higher`	The 'higher' threshold. If not 'NULL', genes with a score larger than 'higher' will be included in the gene set with sign +1. At least one of 'lower' and 'higher' must be specified.
`sign.sets`	Logical, indicating whether the 'signed' slot of the generated CMAPCollection should be set to 'TRUE' or 'FALSE'. This parameter should be set to 'FALSE' when the 'element' does not contain information about directionality, e.g. if it is a p-value.
`...`	Any of the additional arguments detailed above.

See Also

CMAPCollection

Examples

```
data(gCMAPData)
assayDataElementNames(gCMAPData)
cmap <- induceCMAPCollection(gCMAPData, element="z", lower=-2, higher=2)
cmap
notes(cmap)
```

KEGG2cmap *Functions to generate species-specific CMAPCollections from Bioconductor KEGG.db, reactome.db or GO.db annotation packages or the wikipathways* http://www.wikipathways.org/index.php/Download_Pathways *project.*

Description

These functions extract the gene sets defined for a species of interest from the KEGG, Reactome or GO annotation packages or download the latest gene sets from the Wikipathways website and create a CMAPCollections. Wikipathways provides gene identifiers using different annotation sources; the wiki2cmap function only considers Entrez and Ensembl gene identifiers and return a CMAPCollection with Entrez ids. Please note that the GO graph structure will be used to associate every gene with all upstream annotation terms. The relationships between GO categories is not represented in the output CMAPCollection.

Usage

```
KEGG2cmap(species, annotation.package)
reactome2cmap(species, annotation.package)
wiki2cmap(species, annotation.package)
go2cmap(annotation.package="org.Hs.eg.db", ontology="BP", evidence=NULL)
```

Arguments

species	Character, the identifier used by the KEGG or Reactome databases to identify the species of interest. For example, human categories are identified by 'Homo sapiens' in the Reactome and 'hsa' in the KEGG database.
annotation.package	Character, the name of the Bioconductor annotation package for the species of interest, e.g. 'org.Hs.eg.db' for human gene annotations.
ontology	One of 'BP', 'MF' and 'CC', identifying the 'biological process', 'molecular function' and 'cellular component' gene ontology domain of interest.
evidence	Character string identifying the evidence required for a GO annotation to be included in the result. If 'NULL' all annotated terms will be included.

Value

A CMAPCollection object.

Author(s)

Thomas Sandmann

See Also

induceCMAPCollection

Examples

```
## Not run:
KEGG2.hs <- KEGG2cmap( species="hsa",
                       annotation.package="org.Hs.eg.db")

reactome.hs <- reactome2cmap( species="Homo sapiens",
                              annotation.package="org.Hs.eg.db")
wikipathways.hs <- wiki2cmap( species="Homo sapiens",
                              annotation.package="org.Hs.eg.db")

## End(Not run)
```

---

mapNmerge *A function to map eSet featureNames and calculate summaries for many-to-one mapping features*

Description

This function converts the featureNames of an eSet-derived object, either by applying a user-specified translation function (e.g. to remove pre- or suffices) or by refering to the annotation slot of the object to locate the corresponding Bioconductor annotation package.

In cases where multiple features map to the same target identifier, scores are summarized by applying 'summary.fun' (default: mean). For eSet-like object with multiple assayDataElements, each element is summarized separately.

Usage

```
mapNmerge(eset, translation.fun = NULL, get = "ENTREZID", verbose = FALSE, summary.fun = function(x) mea
```

Arguments

eset	An eSet-like object.
translation.fun	A function that will be applied to the results of applying the 'featureNames' method to the eSet. If not 'NULL', this parameter takes precendence and the 'get' parameter will be ignored.

get	A character vector specifying the gene identifier universe to be retrieved from the Bioconductor annotation package.
verbose	Logical, should basid mapping statistics be returned ?
summary.fun	A function that will be applied to the scores after featureName mapping (default: mean).

Value

An eSet object with the same number of samples as the original and one row for each unique new featureName (after mapping & summary).

Note

For large eSet objects, applying 'summary.fun' can be time-consuming. Other strategies, e.g. based on selecting a single probe for each gene based on cross-sample variability are available in the genefilter package.

Author(s)

Thomas Sandmann, sandmann.thomas@gene.com

Examples

```
## Not run:
## requires hgu95av2.db annotation package

if( require( "hgu95av2.db" )) {
  data(sample.ExpressionSet) ## from Biobase
  dim(sample.ExpressionSet)
  head(featureNames(sample.ExpressionSet))
  entrez <- mapNmerge(sample.ExpressionSet)
  dim(entrez)
  head(featureNames(entrez))
}

## End(Not run)
```

---

matrix_or_big.matrix-class

Class "matrix_or_big.matrix"

Description

Union of base's 'matrix' and bigmemory's 'big.matrix' objects.

Objects from the Class

A virtual Class: No objects may be created from it.

Note

Helper class used to dispatch methods on either matrix or big.matrix objects.

Author(s)

Thomas Sandmann, sandmann.thomas@gene.com

See Also

big.matrix for links to other classes ~

`memorize` *Create a new NChannelSet instance by selecting specific channels and load BigMatrix assayData into memory, if present*

Description

This function converts BigMatrix objects stored in the assayData slot of NChannelSets into standard matrices, loading them fully into memory. Standard matrices are returned unchanged.

Usage

```
memorize(object, names, ...)
```

Arguments

`object`	An NChannelSet object.
`names`	Character vector of named channels (default: all channels are returned).
`...`	Additional arguments.

Value

Instance of the same class as 'object'.

Note

This function can be applied to any class inheriting from the virtual eSet class. For non NChannelSets, meta data may not be transferred correctly.

Author(s)

Thomas Sandmann

See Also

BigMatrix eSet memorize selectChannels

Examples

```
## load ExpressionSet
data("sample.ExpressionSet") ## from Biobase
sample.ExpressionSet ## two assayDataElements: exprs, se

## 'exprs' data matrix
class( assayDataElement( sample.ExpressionSet, "exprs" ) )

## convert assayData to BigMatrix objects
storage.file <- tempfile() ## create path & basemane for BigMatrices
storage.file

on.disk <- eSetOnDisk( sample.ExpressionSet, storage.file )
on.disk ## ExpressionSet
dir(dirname( storage.file )) ## created 3 files per channel

class( assayDataElement( on.disk, "exprs" ) ) ## BigMatrix object

## BigMatrix objects are loaded only upon subsetting
assayDataElement( on.disk, "exprs") ## retrieves BigMatrix, NOT matrix
head( assayDataElement( on.disk, "exprs")[,] ) ## retrieves matrix

## convert back to standard in-memory ExpressionSet
in.memory <- memorize( on.disk ) ## all channels
class( assayDataElement( in.memory, "exprs" ) ) ## matrix object
assayDataElementNames( in.memory )

in.memory <- memorize( on.disk, names="exprs" ) ## channel "exprs" only
assayDataElementNames( in.memory )

## remove tempfiles generated for this example
unlink(paste(storage.file,"*", sep=""))
```

`mergeCMAPs` *This function merged two eSets.*

Description

This function merges two eSet objects, if all of the following conditions are met:

- Both objects 'x' and 'y' have to be instances of the same class.
- 'x' and 'y' must be annotated with the same character string in their 'annotation' slots.
- 'x' and 'y' must have the same AssayDataElementNames / channels.
- 'x' and 'y' must have disctinct sampleNames.
- 'x' and 'y' must have the same varLabels / pData columns.

Usage

```
mergeCMAPs(x, y)
```

Arguments

x	An eSet.
y	An eSet of the same class as 'x'

Value

An eSet of the same class as 'x' and 'y'.

Author(s)

Thomas Sandmann, sandmann.thomas@gene.com

Examples

```
library(Biobase)
data( sample.ExpressionSet)

## Not run:
## this doesn't work, because 'x' and 'y' have identical sampleNames
mergeCMAPs( sample.ExpressionSet, sample.ExpressionSet)

## End(Not run)

y <- sample.ExpressionSet
sampleNames( y ) <- paste( sampleNames( y ), "y", sep=".")
mergeCMAPs( sample.ExpressionSet, y )
```

---

mgsa_score-methods *Model-based gene set analysis (MGSA)*

Description

This method is a wrapper for the mgsa methods from the Bioconductor package mgsa, which must be available on the system for the methods to run. The model-based gene set analysis (MGSA) analyzes all categories at once by embedding them in a Bayesian network, naturally taking overlap between categories into account and avoiding the need for multiple testing correction. Please consult the mgsa help page for more details.

Arguments

query	A CMAPCollection, GeneSet, or GeneSetCollection object containing the 'query' gene sets to compare against the 'sets'
sets	A CMAPCollection, GeneSetCollection or GeneSet object
universe	A character string of gene ids for all genes that could potentially be of interest, e.g. all genes represented on a microarray, all annotated genes, etc.

`keep.scores` Logical: store the identifiers for the genes detected in 'query' and 'sets' ? (Default: FALSE) The size of the generated CMAPResults object increases with the number of contained gene sets. For very large collections, setting this parameter to 'TRUE' may require large amounts of memory.

`element` A character string corresponding to the assayDataElementName of the NChannelSet object to be thresholded on the fly with the `induceCMAPCollection`.

`lower` The lower threshold for the `induceCMAPCollection`.

`higher` The 'higher' threshold for the `induceCMAPCollection`.

`min.set.size` Number of genes a gene set induced by `induceCMAPCollection` needs to contain to be included in the analysis (Default:5).

`...` Additional arguments passed to mgsa function from the mgsa package, including the following:

- alpha: Grid of values for the parameter alpha. Values represent probabilities of false-positive events and hence must be in [0,1]. numeric
- beta: Grid of values for the parameter beta. Values represent probabilities of false-negative events and hence must be in [0,1]. numeric.
- p: Grid of values for the parameter p. Values represent probabilities of term activity and therefore must be in [0,1]. numeric.
- steps: The number of steps of each run of the MCMC sampler. integer of length 1. A recommended value is 1e6 or greater.
- restarts: The number of different runs of the MCMC sampler. integer of length 1. Must be greater or equal to 1. A recommended value is 5 or greater.
- threads: The number of threads that should be used for concurrent restarts. A value of 0 means to use all available cores. Defaults to 'getOption(mc.cores, default=0)', which will instruct mgsa to use all available cores.

Value

A CMAPResults object. The reported p-values represent '1-marginal posterior probability'. For the 'effect' column, the p-values have been transformed to z-scores using a standard normal distribution.

Methods

```
signature(query = "GeneSet", sets = "CMAPCollection", universe = "character")
signature(query = "GeneSet", sets = "NChannelSet", universe = "character")
signature(query = "SignedGeneSet", sets = "CMAPCollection", universe = "character")

signature(query = "SignedGeneSet", sets = "NChannelSet", universe = "character")

signature(query = "GeneSetCollection", sets = "CMAPCollection", universe = "character")

signature(query = "GeneSetCollection", sets = "NChannelSet", universe = "character")
```

```
signature(query = "GeneSetCollection", sets = "GeneSetCollection", universe = "character")

signature(query = "GeneSet", sets = "GeneSetCollection", universe = "character")

signature(query = "GeneSet", sets = "GeneSet", universe = "character")
signature(query = "CMAPCollection", sets = "CMAPCollection", universe = "character")

signature(query = "CMAPCollection", sets = "GeneSetCollection", universe = "character")
```

Note

This Bayesian approach does not require any additional correction of p-values for multiple testing. For consistency, the returned CMAPResults object contains a padj column duplicating the content of the pval column.

See Also

mgsa

Examples

```
if( is.element("mgsa", installed.packages()[,1])){
   require( "mgsa", character.only = TRUE )

   data(gCMAPData)
   gene.set.collection <- induceCMAPCollection(gCMAPData, "z",
   higher=2, lower=-2)

    ## compare all gene sets in the gene.set.collection
    ## to each other
    universe = featureNames(gCMAPData)
     mgsa_score(gene.set.collection, gene.set.collection,
     universe = universe)
  }
```

---

minSetSize-methods *GeneSetCollection length filtering*

Description

This function filteres a GeneSetCollection by removing all contained GeneSets that do not include at least the user-specified number of genes also found in the user-specified universe.

Usage

```
## S4 method for signature 'CMAPCollection'
minSetSize(sets, universe=NULL, min.members = 5)
```

Arguments

sets	A CMAPCollection object.
universe	Optional character vector of gene identfiers to be considered as the universe. Only geneIds included in the universe will count toward the gene set membership counts. If 'NULL' (default), all featureNames of the CMAPCollection will be considered.
min.members	Number of genes (in the univerese) a gene set needs to contain to be retained.

Value

A CMAPCollection with all gene sets containing more than the specified number of members.

Author(s)

Thomas Sandmann

Examples

```
data(gCMAPData)
gene.set.collection <- induceCMAPCollection(gCMAPData, "z", higher=2, lower=-2)

minSetSize(gene.set.collection, min.members=100)
```

---

mroast_score-methods *Methods for Function* mroast_score *in Package* **gCMAP**

Description

These methods provide a wrapper for the Rotation Gene Set Tests function mroast mroast tests whether any of the genes in the set are differentially expressed.

Usage

```
## S4 method for signature 'eSet,CMAPCollection'
mroast_score(experiment,sets,predictor=NULL, design.matrix=NULL,element="exprs", keep.scores=FALSE,.

## S4 method for signature 'matrix,CMAPCollection'
mroast_score(experiment, sets,...)

## S4 method for signature 'matrix,GeneSet'
mroast_score(experiment,sets,...)

## S4 method for signature 'eSet,GeneSet'
mroast_score(experiment, sets,...)

## S4 method for signature 'matrix,GeneSetCollection'
```

```
mroast_score(experiment,sets,...)

## S4 method for signature 'eSet,GeneSetCollection'
mroast_score(experiment,sets,...)
```

Arguments

sets	A CMAPCollection, GeneSetCollection or GeneSet object containing gene sets, with which to query the experiment object.
experiment	An eSet or data matrix with numeric data to compare the query object to.
predictor	A character vector or factor indicating the phenotypic class of the experiment data columns. Either the 'predictor' or 'design' parameter must be supplied.
design.matrix	A design matrix for the experiment. Either the 'predictor' or 'design' parameter must be supplied. If both are supplied, the 'design' is used.
element	Character vector specifying which channel of an eSet to extract (defaults to "exprs", alternatives may be e.g. "z", etc.)
keep.scores	Logical: keep gene-level scores for all gene sets (Default: FALSE) ? The size of the generated CMAPResults object increases with the number of contained gene sets. For very large collections, setting this parameter to 'TRUE' may require large amounts of memory.
...	Additional arguments passed to downstream methods.

Value

A CMAPResults object.

References

Goeman, JJ, and Buhlmann, P (2007). Analyzing gene expression data in terms of gene sets: methodological issues. Bioinformatics 23, 980-987.

Langsrud, O (2005). Rotation tests. Statistics and Computing 15, 53-60.

Phipson B, and Smyth GK (2010). Permutation P-values should never be zero: calculating exact P-values when permutations are randomly drawn. Statistical Applications in Genetics and Molecular Biology, Volume 9, Article 39.

Routledge, RD (1994). Practicing safe statistics with the mid-p. Canadian Journal of Statistics 22, 103-110.

Wu, D, Lim, E, Francois Vaillant, F, Asselin-Labat, M-L, Visvader, JE, and Smyth, GK (2010). ROAST: rotation gene set tests for complex microarray experiments. Bioinformatics 26, 2176-2182. http://bioinformatics.oxfordjournals.org/cgi/content/abstract/btq401?

Examples

```
data(gCMAPData)
gene.set.collection <- induceCMAPCollection(gCMAPData, "z", higher=2, lower=-2)
sampleNames( gene.set.collection ) <- c("set1", "set2", "set3")

## random score matrix
```

```
y <- matrix(rnorm(1000*6),1000,6, dimnames=list(featureNames(gCMAPData), 1:6))
## set1 is differentially regulated
effect <- as.vector(members(gene.set.collection[,1]) * 2)
y[,4:6] <- y[,4:6] + effect

predictor <- c( rep("Control", 3), rep("Case", 3))

res<- mroast_score(y, gene.set.collection, predictor = predictor,keep.scores=TRUE)
res

## heatmap of expression scores for set1
set1.expr <- geneScores(res)[["set1"]]
heatmap(set1.expr, scale="none", Colv=NA, labCol=predictor,
        RowSideColors=ifelse( attr(set1.expr, "sign") == "up", "red", "blue"),
        margin=c(7,5))
legend(0.35,0,legend=c("up", "down"), fill=c("red", "blue"), title="Annotated sign", horiz=TRUE, xpd=TRUE)
```

`pairwise_compare` *Generate statistics associated with pairwise differential expression*

Description

When provided with an ExpressionSet, comparisons are made between control and perturbation samples.

Usage

```
pairwise_compare(eset, control_perturb_col = "cmap", control="control", perturb="perturbation")
pairwise_compare_limma(eset, control_perturb_col = "cmap",
control="control", perturb="perturbation", limma.index=2)
```

Arguments

`eset`	ExpressionSet with all array data for a single instance, plus metadata on which arrays are perturbation and control.
`control_perturb_col`	Column name in `phenoData` of `eset` where control/perturbation designations are stored.
`control`	String designating control samples in the `control_perturb_col` column.
`perturb`	String designating perturbation samples in the `control_perturb_col` column.
`limma.index`	Integer specifying the index of the parameter estimate for which we to extract t and other statistics. The default corresponds to a two-class comparison with the standard parameterization. The function assumes that there was no missing data, so that test for all genes were performed on the same sample size.

Value

The function returns a data frame with the following columns:

log_fc	Log fold-change between perturbed and control data. (A positive value denotes higher expression in the perturbed samples.)
z	When at least one condition has two or more samples, the pairwise_compare_limma functions uses lmFit, eBayes and topTable to compare the two classes and compute an (uncorrected) limma p-value. The pairwise_compare functions performs a standard t-test instead. For ease of comparison across instances with different numbers of samples, either p-value is converted to the standard normal scale. The result is reported here. As for fc, positive values denote higher expression in perturbed samples.
p	When at least one condition has two or more samples, the two-tailed standard (pairwise_compare) or **limma** p-value (pairwise_compare_limma), as computed by eBayes. Note that this p-value can also be computed from z, via pnorm (doubling for two tails).

Note

The pairwise_compare functions returns p-values from a standard t-test. The pairwise_compare_limma functions uses the **limma** package instead to perform a moderated t-test.

---

pairwise_DESeq *Generate statistics associated with pairwise differential expression from RNAseq count data*

Description

When provided with an CountDataSet, comparisons are made between control and perturbation samples.

Usage

```
 pairwise_DESeq(cds, vst, control_perturb_col = "condition",
control="control", perturb="perturbation", try.hard=FALSE)
```

Arguments

cds	CountDataSet with all count data for a single instance, plus metadata on which samples are perturbation and control.
vst	Matrix of variance-stabilized count data that must include columns with colnames matching the sampleNames of the cds object. The vst matrix may contain additional columns / samples, which will be ignored.
control_perturb_col	Column name in phenoData of cds where control/perturbation designations are stored.

control	String designating control samples in the control_perturb_col column.
perturb	String designating perturbation samples in the control_perturb_col column.
try.hard	Logical parameter indicating how to proceed when DESeq's parametric estimation of the dispersion parameter fails. If set to FALSE (default), the function exits with an error. If set to TRUE, the function will try a non-parametric approach instead.

Value

The function returns a data frame with the following columns:

log_fc	Moderated log2 fold-change between perturbed and control data. (A positive value denotes higher expression in the perturbed samples.) The change was calculated from the (mean) counts after variance stabilizing transformation. Please consult the **DESeq** vignette for details on the transformation.
z	For ease of comparison across instances with different numbers of samples, the (uncorrected) DESEq p-value is converted to the standard normal scale. The result is reported here. As for log_fc, positive values denote higher expression in perturbed samples.
p	p-value for differential expression calculated by the nbinomTest function from the **DESeq** package. In the absence of replicates, the dispersion parameter is estimated across all samples, ignoring the class labels, by using the blind method of the estimateDispersions function. When replicates are available, the pooled method is used instead. Note that this p-value can also be computed from z, via pnorm (doubling for two tails).

Note

To use this function, please install the suggested Bioconductor package 'DESeq'.

---

romer_score-methods *Methods for Function* romer_score *in Package* **gCMAP**

Description

These methods provide a wrapper for the Rotation Gene Set Enrichment Analysis function romer Romer performes a competitive test in that the different gene sets are pitted against one another. Instead of permutation, it uses rotation, a parametric resampling method suitable for linear models (Langsrud, 2005).

Usage

```
## S4 method for signature 'eSet,CMAPCollection'
romer_score(experiment,sets,predictor=NULL,
design.matrix=NULL, element="exprs", keep.scores=FALSE, ...)
```

```
## S4 method for signature 'matrix,CMAPCollection'
romer_score(experiment, sets,...)

## S4 method for signature 'matrix,GeneSet'
romer_score(experiment,sets,...)

## S4 method for signature 'eSet,GeneSet'
romer_score(experiment, sets,...)

## S4 method for signature 'matrix,GeneSetCollection'
romer_score(experiment,sets,...)

## S4 method for signature 'eSet,GeneSetCollection'
romer_score(experiment, sets,...)
```

Arguments

sets	A CMAPCollection, GeneSetCollection or GeneSet object containing gene sets, with which to query the experiment object.
experiment	An eSet or data matrix with numeric data to compare the query object to.
predictor	A character vector or factor indicating the phenotypic class of the experiment data columns. Either the 'predictor' or 'design' parameter must be supplied.
design.matrix	A design matrix for the experiment. Either the 'predictor' or 'design' parameter must be supplied. If both are supplied, the 'design' is used.
element	Character vector specifying which channel of an eSet to extract (defaults to "exprs", alternatives may be e.g. "z", etc.)
keep.scores	Logical: keep gene-level scores for all gene sets (Default: FALSE) ? The size of the generated CMAPResults object increases with the number of contained gene sets. For very large collections, setting this parameter to 'TRUE' may require large amounts of memory.
...	Additional arguments passed to downstream methods.

Value

A CMAPResults object.

References

Langsrud, O, 2005. Rotation tests. Statistics and Computing 15, 53-60

Doerum G, Snipen L, Solheim M, Saeboe S (2009). Rotation testing in gene set enrichment analysis for small direct comparison experiments. Stat Appl Genet Mol Biol, Article 34.

Majewski, IJ, Ritchie, ME, Phipson, B, Corbin, J, Pakusch, M, Ebert, A, Busslinger, M, Koseki, H, Hu, Y, Smyth, GK, Alexander, WS, Hilton, DJ, and Blewitt, ME (2010). Opposing roles of polycomb repressive complexes in hematopoietic stem and progenitor cells. Blood, published online 5 May 2010. http://www.ncbi.nlm.nih.gov/pubmed/20445021

Subramanian, A, Tamayo, P, Mootha, VK, Mukherjee, S, Ebert, BL, Gillette, MA, Paulovich, A, Pomeroy, SL, Golub, TR, Lander, ES and Mesirov JP, 2005. Gene set enrichment analysis: a

knowledge-based approach for interpreting genome-wide expression profiles. Proc Natl Acad Sci U S A 102, 15545-15550

Examples

```
data(gCMAPData)
gene.set.collection <- induceCMAPCollection(gCMAPData, "z", higher=2, lower=-2)
sampleNames( gene.set.collection ) <- c("set1", "set2", "set3")

## random score matrix
y <- matrix(rnorm(1000*6),1000,6, dimnames=list(featureNames(gCMAPData), 1:6))

## set1 is differentially regulated
effect <- as.vector(members(gene.set.collection[,1]) * 2)
y[,4:6] <- y[,4:6] + effect

predictor <- c( rep("Control", 3), rep("Case", 3))

res <- romer_score(y, gene.set.collection, predictor = predictor, keep.scores=TRUE)
res

## heatmap of expression scores for set1
set1.expr <- geneScores(res)[["set1"]]
heatmap(set1.expr, scale="none", Colv=NA, labCol=predictor,
        RowSideColors=ifelse( attr(set1.expr, "sign") == "up", "red", "blue"),
        margin=c(7,5))
legend(0.35,0,legend=c("up", "down"), fill=c("red", "blue"), title="Annotated sign", horiz=TRUE, xpd=TRUE)
```

`SignedGeneSet` *Construtor for SignedGeneSet*

Description

The construtor is largely identical to `GeneColorSet`, but also handles an optional `geneSign` argument, which is an alias for `geneColor`.

Methods

`signature(type = "ANY")` Construtor which uses a template object. See all methods for the `GeneColorSet` constructor. If a `geneSign` argument is included by name, it will be used to populate the `geneColor` slot of the returned object.

`signature(type = "missing")` Basic method with no template object.

SignedGeneSet-class *Class* "SignedGeneSet"

Description

A simple extension of GeneColorSet which forces geneColor to be either "down" or "up" and which ignores phenotype and phenotypeColor slots.

Objects from the Class

Construct a SignedGeneSet with the SignedGeneSet constructor method, or with a call to new. Although SignedGeneSet derives from the more abstract GeneColorSet, not phenotype argument is required; if phenotype is supplied (or is present in a template object), it will be ignored.

Slots

See GeneColorSet. No additional slots are added.

Extends

Class "GeneColorSet", directly. Class "GeneSet", by class "GeneColorSet", distance 2.

Methods

Methods specific to SignedGeneSet:

downIds signature(object = "SignedGeneSet"): retrieve geneIds entires for which geneSign == "down".

geneSign signature(obj = "SignedGeneSet"): alias for geneColor slot.

geneSign<- signature(object = "SignedGeneSet", value = "character"): alias for geneColor slot, converting to factor automatically.

geneSign<- signature(object = "SignedGeneSet", value = "factor"): alias for geneColor slot.

initialize signature(.Object = "SignedGeneSet"): on construction, checks for appropriate geneSign/geneColor values and sets phenotype and phenotypeColor to empty strings, since these are ignored. If no geneSign/geneColor values are supplied, "up" will be used by default.

show signature(object = "SignedGeneSet"): same as for GeneColorSet but suppresses display of unused phenotype and phenotypeColor slots.

upIds signature(object = "SignedGeneSet"): retrieve geneIds entires for which geneSign == "up".

mapIdentifiers signature(object = "SignedGeneSet"): Extends the 'mapIdentifiers' method implemented for GeneSets in the GSEABase package, but rejects target gene ids when multiple different (probe) identifiers with different gene signs (up / down) map to the same target.

incidence signature(object = "SignedGeneSet") and

incidence signature(object = "GeneSetCollection"): Mirror the 'incidence' method implemented for GeneSets in the GSEABase package, but returns sparseMatrix objects containing -1 / +1 to indicate up- and down-regulated gene members.

Examples

```
gene.ids <- letters[1:10]
gene.signs <- rep(c("up","down"), each=5)
SignedGeneSet(gene.ids, geneSign=gene.signs, setName="set1")
```

`signedRankSumTest` *An implementation of the Wilcox rank sum test / Mann-Whitney test that takes into account the direction / sign of gene set members and possibly the correlation between cases*

Description

This test evaluates whether the mean rank of statistics of gene set members is greater or less than the mean rank of the remaining statistic values. It extends the rankSumTestWithCorrelation function from the 'limma' package by taking into account the 'sign' of gene set members by reversing the ranks of down-regulated genes.

Usage

```
signedRankSumTest(statistics, index.up, index.down = NULL,
input.is.ranks=FALSE, correlation=0, df = Inf, adjust.ties=TRUE)
```

Arguments

`statistics`	numeric vector giving values of the test statistic.
`index.up`	an index vector such that statistics[index.up] contains the values of the statistic for the up-regulated genes.
`index.down`	an index vector such that statistics[index.down] contains the values of the statistic for the down-regulated genes.
`correlation`	numeric scalar, average correlation between cases in the test group. Cases in the second group are assumed independent of each other and the first group.
`df`	degrees of freedom which the correlation has been estimated.
`adjust.ties`	logical: correct for ties ?
`input.is.ranks`	logical: is 'statistics' a vector of ranks ? If FALSE (default), ranks are computed. If FALSE, 'statistics' is assumed to represent ranks and is used directly.

Details

Please see the rankSumTestWithCorrelation function from the limma package for details.

Value

Numeric vector containing U-statistic, z-score and p-value.

Author(s)

Thomas Sandmann

References

Wu, D, and Smyth, GK (2012). Camera: a competitive gene set test accounting for inter-gene correlation. Submitted.

Barry, W.T., Nobel, A.B., and Wright, F.A. (2008). A statistical framework for testing functional categories in microarray data. Annals of Applied Statistics 2, 286-315.

Zar, JH (1999). Biostatistical Analysis 4th Edition. Prentice-Hall International, Upper Saddle River, New Jersey.

See Also

rankSumTestWithCorrelation

Examples

```
genes.up   <- c(1:10)
genes.down <- c(21:30)

set.seed(123)
scores <- matrix(rnorm(200), ncol=2)

## the first gene set receives increased /
## decreased scores in the first experiment
scores[genes.up,1]   <- scores[genes.up  ,1] + 1
scores[genes.down,1] <- scores[genes.down,1] - 1

## significantly greater
signedRankSumTest( statistics = scores[,1],
                   index.up   = genes.up,
                   index.down = genes.down)

## not signficant
signedRankSumTest( statistics = scores[,2],
                   index.up   = genes.up,
                   index.down = genes.down)
```

`splitPerturbations` *Function to split an ExpressionSet downloaded from ArrayExpress based on the experimental factors present in the phenoData slot*

Description

The ArrayExpress Bioconductor package provides access to microarray and RNAseq datasets from the EBI's ArrayExpress repository. Sample and experiment annotations are contained in the phenoData slot and can be used to automatically construct single-factor comparisons by subsetting the original ExpressionSet object. This function be used to automatically identify perturbation vs. control comparisons and splits the original dataset in to instance-level objects.

Usage

```
splitPerturbations(eset, control = "none", controlled.factors = NULL, factor.of.interest = "Compound",
```

Arguments

`eset`	An eSet object with experimental factors annotated in the phenoData slot. Experimental factors are identified by the prefix of the column name, specified in the 'prefix' parameter. In ExpressionSets obtained from the ArrayExpress respository experimental factors can be identified by their "^Factor" prefix.
`control`	A character string identifying control samples in the 'factor.of.interest' column.
`controlled.factors`	A character vector specifying which annotation columns should be matched to assign controls to perturbation samples. If the set to NULL shared controls are used for all perturbations. If set to 'all' all experimental factors must be identical between control and perturbation samples. Alternatively, individual factors and their combinations can be specified as a vector, e.g. as c("Vehice", "Time"). Column names can be abbreviated as long as they uniquely identify the pData column.
`factor.of.interest`	Character string, the name of the pData column containing the factor of interest. Column name can be abbreviated aslong as it uniquely identifies the pData column.
`ignore.factors`	A character vector with valid pData names specifying annotation columns to exclude. Column names can be abbreviated as long as they uniquely identify the pData column.
`cmap.column`	Column name for an additional annotation column that will be added to all generated eSets. Used by the `generate_gCMAP_NChannelSet` function.
`prefix`	String identifying pData columns with experimental factors. Setting the prefix to NULL will include all pData columns.

Details

To identify 'perturbation versus control' comparisons, the user needs to inspect the phenoData slot (see examples) and choose the appropriate factor of interest as well as a term in this column that identifies experimental control samples. The 'controlled.factors' character string identifies additional factors (= columns in the phenoData slot), in which control samples must match their corresponding perturbation samples.

For example (see example code section), an ExpressionSet may be annotated with two different annotated factors, Compound and Solven, corresponding to two columns in the pData slot.

The first column is of interest and therefore 'factor.of.interest should be set to 'Compound'. For each level of 'factor.of.interest' unique experimental conditions are identified based on the remaining pData columns. (To exclude columns, use the 'ignore.factors' parameter.) Separate ExpressionSet objects will be constructed for each unique experimental condition.

To distinguish control samples from perturbations, the 'control' parameter needs to be provided. For example, if control samples in the 'factor.of.interest' column are annotated as 'vehicle', the 'control' parameter should be set to 'vehicle'.

The second column in this example,'Solvent', contains additional information about the type of vehicle used for each experiment, e.g. DMSO, ethanol, etc. To ensure that each sample is matched to the correct control condition the 'controlled.factors' parameter is set to 'Vehicle' to include this annotation column when assigning control to perturbation samples.

To consider all available annotation columns to match controls, the 'controlled.factors' parameters can be set to 'all' instead. (In this example, either setting the parameter to 'Vehicle' or 'all' yields identical results, as there is only one column in addition to the 'factor.of.interest'.)

Value

A list of eSet objects, one for each unique experimental perturbation with perturbation and control samples.

Warning

Annotations for experiments in public repositories are provided by the uploader and vary widely in quality and notation. This function is expected to handle experiments with clear perturbation / control annotations. Mileage on other datasets may vary.

Note

Only experimental instances with valid controls will be returned.

Author(s)

Thomas Sandmann, sandmann.thomas@gene.com

See Also

generate_gCMAP_NChannelSet annotate_eset_list

Examples

```
require(Biobase)
data( sample.ExpressionSet )
head(pData( sample.ExpressionSet))
eset.list <- splitPerturbations( eset=sample.ExpressionSet,
                    factor.of.interest="type",
                    control="Control",
                    controlled.factors="sex",
                    ignore.factors="score",
                    prefix=""
)
```

```
length( eset.list )
## the first eset contains male Cases & controls
pData( eset.list[[1]])
## the second eset contains female Cases & controls
pData( eset.list[[2]])

## generate data.frame with sample annotations
annotate_eset_list( eset.list)
```

---

wilcox_score-methods *Methods for Function* wilcox_score *in Package* **gCMAP**

Description

These methods provide a wrapper for the Mean-rank Gene Set Test function wilcoxGST

wilcox_score is a synonym for gst_score with ranks.only=TRUE. This test procedure was developed by Michaud et al (2008), who called it mean-rank gene-set enrichment.

Usage

```
## S4 method for signature 'matrix,CMAPCollection'
wilcox_score(experiment, sets, adjust.ties=FALSE, keep.scores=FALSE, ...)

## S4 method for signature 'numeric,CMAPCollection'
wilcox_score(experiment, sets,...)

## S4 method for signature 'eSet,CMAPCollection'
wilcox_score(experiment, sets, element="z",...)

## S4 method for signature 'matrix,GeneSet'
wilcox_score(experiment, sets,...)

## S4 method for signature 'numeric,GeneSet'
wilcox_score(experiment, sets,...)

## S4 method for signature 'eSet,GeneSet'
wilcox_score(experiment, sets, element="z",...)

## S4 method for signature 'matrix,GeneSetCollection'
wilcox_score(experiment, sets,...)

## S4 method for signature 'numeric,GeneSetCollection'
wilcox_score(experiment, sets,...)

## S4 method for signature 'eSet,GeneSetCollection'
wilcox_score(experiment, sets, element="z",...)
```

```
## S4 method for signature 'CMAPCollection,eSet'
wilcox_score(experiment, sets, element="z",adjust.ties=FALSE, keep.scores=FALSE,...)

## S4 method for signature 'CMAPCollection,numeric'
wilcox_score(experiment, sets,...)

## S4 method for signature 'CMAPCollection,matrix'
wilcox_score(experiment, sets,...)

## S4 method for signature 'GeneSet,numeric'
wilcox_score(experiment, sets,...)

## S4 method for signature 'GeneSet,matrix'
wilcox_score(experiment, sets,...)

## S4 method for signature 'GeneSet,eSet'
wilcox_score(experiment, sets,element="z",...)

## S4 method for signature 'GeneSetCollection,numeric'
wilcox_score(experiment, sets,...)

## S4 method for signature 'GeneSetCollection,matrix'
wilcox_score(experiment, sets,...)

## S4 method for signature 'GeneSetCollection,eSet'
wilcox_score(experiment, sets,element="z",...)
```

Arguments

sets	A CMAPCollection, GeneSetCollection or GeneSet object containing gene sets, with which to query the experiment object.
experiment	An eSet or matrix or vector with numeric data to compare the query object to.
element	Character vector specifying which channel of an eSet to extract (defaults to "exprs", alternatives may be e.g. "z", etc.)
...	Additional arguments passed on to downstream methods.
adjust.ties	Logical: adjust Wilcox-Mann-Whitney statistic in the presence of ties ? (Default: FALSE)
keep.scores	Logical: keep gene-level scores for all gene sets (Default: FALSE) ? The size of the generated CMAPResults object increases with the number of contained gene sets. For very large collections, setting this parameter to 'TRUE' may require large amounts of memory.

Examples

```
data(gCMAPData)
gene.set.collection <- induceCMAPCollection(gCMAPData, "z", higher=2, lower=-2)
```

```
profile <- assayDataElement(gCMAPData[,1], "z")
## one profile versus three sets
wilcox_score(profile, gene.set.collection)

## three sets versus three profiles
wilcox_score(gene.set.collection, gCMAPData)
```

---

zScores *Function to calculate z-scores from p-values*

Description

Function to calculate z-score from a normal distribution from a two-tailed p-value and sign vector (e.g. log2 fold change). To avoid -Inf/Inf z-scores, p-values < 'limit' are set to 'limit'.

Usage

```
zScores(pval, direction=NULL, tails=2, limit=.Machine$double.xmin)
```

Arguments

pval	Vector with p-values
direction	Vector that will be used to determine the sign of the z-scores. Only the sign of the values is considered, so any suitable vectors (e.g. log2 fold change) can be supplied.
limit	Numeric (default: .Machine$double.xmin). pvalues < 'limit' will be set to 'limit' to avoid Inf/-Inf z-scores. Set to NULL to disable.
tails	Numeric, either 1 for p-values from one-tailed or 2 for p-values from two-tailed tests.

Value

A vector of z-scores

Author(s)

Thomas Sandmann

See Also

qnorm

Index