HTSanalyzeR

April 20, 2011

`aggregatePvals` *Aggregate p-values from gene set over-representation tests.*

Description

This function takes as input a matrix of p-values for example obtained from a GSEA on multiple phenotypes, with a row for each gene set and a column for each phenotype and aggregates the p-values by row (i.e. one aggregated p-value for each gene set) according to Fisher or Stouffer's methods.

Usage

```
aggregatePvals(pvalMatrix, method="fishers", pAdjustMethod="BH",
order=TRUE)
```

Arguments

`pvalMatrix`	a numeric matrix of p-values, with rows named according to the gene set (rows correspond to gene sets, and columns to multiple p-values to be aggregated for that gene set)
`method`	a single character value of "stouffers" or "fishers"
`pAdjustMethod`	a single character value specifying the p-value adjustment method to be used (see 'p.adjust' for details)
`order`	a single logical value: TRUE or FALSE. If it is TRUE, the results table will be ordered according to the aggregated p-values.

Details

The Fisher method combines the p-values into an aggregated chi-squared statistic equal to -2*sum(log(Pk)) were we have k=1,..,K pvalues independently distributed as uniform on the unit interval under the null hypothesis. The resulting p-values are calculated by comparing this chi-squared statistic to a chi-squared distribution with 2K degrees of freedom. The Stouffer method computes a z-statistics assuming that the sum of the quantiles (from a standard normal distribution) corresponding to the p-values are distributed as N(0,K).

Value

a matrix with a row for each gene set and two columns: "Aggregated.p.value" and "Adjusted.aggregated.p.value"

Author(s)

Jack Rose and Camille Terfve

Examples

```
p1 <- runif(100, min=0, max=1)
p2 <- runif(100, min=0, max=1)
names(p1) <- as.character(sample(x=seq(from=1, to=100, by=1), size=100,
replace=FALSE))
pmatrix <- cbind(p1, p2)
rownames(pmatrix) <- names(p1)
aggP <- aggregatePvals(pvalMatrix=pmatrix, method="stouffers")
```

---

analyzeGeneSetCollections

Hypergeometric tests and Gene Set Enrichment Analyses over a list of gene set collections

---

Description

This function takes a list of gene set collections, a named phenotype vector (with names of the phenotype vector as the universe), a vector of hits (gene names only) and returns the results of hypergeometric and gene set enrichment analyses for all of the gene set collections (with multiple hypothesis testing corrections).

Usage

```
analyzeGeneSetCollections(listOfGeneSetCollections, geneList, hits,
pAdjustMethod="BH", pValueCutoff=0.05, nPermutations=1000,
minGeneSetSize=15, exponent=1, verbose=TRUE)
```

Arguments

`listOfGeneSetCollections`
: a list of gene set collections (a 'gene set collection' is a list of gene sets). Even if only one collection is being tested, it must be entered as an element of a 1-element list, e.g. `ListOfGeneSetCollections = list(YourOneGeneSetCollectior` Naming the elements of listOfGeneSetCollections will result in these names being associated with the relevant data frames in the output (meaningful names are advised)

`geneList`
: a numeric or integer vector of phenotypes in descending or ascending order with elements named by their EntrezIds (no duplicates nor NA values)

`hits`
: a character vector of the EntrezIds of hits, as determined by the user

`pAdjustMethod`
: a single character value specifying the p-value adjustment method to be used (see 'p.adjust' for details)

`pValueCutoff`
: a single numeric value specifying the cutoff for p-values considered significant

`nPermutations`
: a single integer or numeric value specifying the number of permutations for deriving p-values in GSEA

`minGeneSetSize`
: a single integer or numeric value specifying the minimum number of elements in a gene set that must map to elements of the gene universe. Gene sets with fewer than this number are removed from both hypergeometric analysis and GSEA.

`exponent`
: a single integer or numeric value used in weighting phenotypes in GSEA (see the function `gseaScores`)

`verbose`
: a single logical value specifying to display detailed messages (when verbose=TRUE) or not (when verbose=FALSE)

Details

All gene names must be EntrezIds in 'listOfGeneSetCollections', 'geneList', and 'hits'.

Value

`HyperGeo.results`
: a list of data frames containing the results for all gene set collections in the input.

`GSEA.results`
: a similar list of data frames containing the results from GSEA. As an example, to access the GSEA results for a gene set collection named "MyGeneSetCollection", one would enter: output$GSEA.results$MyGeneSetCollection

`Sig.pvals.in.both`
: a list of data frames containing the gene sets with p-values considered significant in both hypergeometric test and GSEA, before p-value correction. Each element of the list contains the results for one gene set collection.

`Sig.adj.pvals.in.both`
: a list of data frames containing the gene sets with p-values considered significant in both hypergeometric test and GSEA, after p-value correction. Each element of the list contains the results for one gene set collection.

Author(s)

John C. Rose, Xin Wang

See Also

`analyze`

Examples

```
## Not run:
library(org.Dm.eg.db)
library(GO.db)
library(KEGG.db)
##load phenotype vector (see the vignette for details about the
##preprocessing of this data set)
data("KcViab_Data4Enrich")
##Create a list of gene set collections for Drosophila melanogaster (Dm)
GO_MF <- GOGeneSets(species="Dm", ontologies="MF")
PW_KEGG <- KeggGeneSets(species="Dm")
ListGSC <- list(GO_MF=GO_MF, PW_KEGG=PW_KEGG)
##Conduct enrichment analyses
GSCAResults <- analyzeGeneSetCollections(
listOfGeneSetCollections=ListGSC,
geneList=KcViab_Data4Enrich,
```

```
hits=names(KcViab_Data4Enrich)[which(abs(KcViab_Data4Enrich)>2)],
pAdjustMethod="BH",
nPermutations=1000,
minGeneSetSize=200,
exponent=1,
verbose=TRUE
)

## End(Not run)
```

`analyze` *Gene Set Collection Analysis or NetWork Analysis*

Description

This is a generic function.

When implemented as the S4 method for objects of class `GSCA`, this function invokes function `analyzeGeneSetCollections` to do hypergeometic tests and GSEA.

When implemented on an object of class `NWA`, it calls function `networkAnalysis` to do subnetwork identification.

To use this function for objects of class `GSCA`:

analyze(object, para = list(pValueCutoff = 0.05, pAdjustMethod = "BH", nPermutations = 1000, minGeneSetSize = 15, exponent = 1), verbose = TRUE)

To use this function for objects of class `NWA`:

analyze(object, fdr=0.001, species, verbose=TRUE)

Usage

```
analyze(object, ...)
```

Arguments

`object`	an object. When this function is implemented as the S4 method of class 'GSCA' or 'NWA', this argument is an object of class 'GSCA' or 'NWA'.
`...`	other arguments depending on class (see below for the arguments supported by the method of class 'GSCA' or 'NWA')

para: a list of parameters for GSEA and hypergeometric tests including:

para$pValueCutoff a single numeric value specifying the cutoff for p-values considered significant

para$pAdjustMethod: a single character value specifying the p-value adjustment method to be used (see 'p.adjust' for details)

para$nPermutations: a single integer or numeric value specifying the number of permutations for deriving p-values in GSEA

para$minGeneSetSize: a single integer or numeric value specifying the minimum number of elements in a gene set that must map to elements of the gene universe. Gene sets with fewer than this number are removed from both hypergeometric analysis and GSEA.

para$exponent: a single integer or numeric value used in weighting phenotypes in GSEA (see 'gseaScores' function)

fdr: a single numeric value specifying the false discovery for the scoring of nodes (see `BioNet::scoreNodes` and Dittrich et al., 2008 for details)

species: a single character value specifying the species for which the data should be read. The current version supports one of the following species: "Dm" ("Drosophila_melanogaster"), "Hs" ("Homo_sapiens"), "Rn" ("Rattus_norvegicus"), "Mm" ("Mus_musculus"), "Ce" ("Caenorhabditis_elegans"). This is an optional argument here. If it is provided, then the labels of nodes of the identified subnetwork will be mapped from Entrez IDs to gene symbols; otherwise, Entrez IDs will be used as labels for those nodes.

verbose: a single logical value specifying to display detailed messages (when verbose=TRUE) or not (when verbose=FALSE)

Details

For objects of class `GSCA`:

The function will store the results from function `analyzeGeneSetCollections` in slot `result`, and update information about these results to slot `summary` of class `GSCA`.

See function `analyzeGeneSetCollections` for the detailed information about the returned results.

For objects of class `NWA`:

The function will store the subnetwork module identified by BioNet (if `species` is given, labels of nodes will also be mapped from Entrez IDs to gene symbols), and update information about these results to slot `summary` of class `NWA`.

See function `networkAnalysis` for the detailed information about the returned results.

Value

In the end, this function will return an updated object of class `GSCA` or `NWA`.

Author(s)

Xin Wang <xw264@cam.ac.uk>

References

Beisser D, Klau GW, Dandekar T, Muller T, Dittrich MT. *BioNet: an R-Package for the functional analysis of biological networks.* Bioinformatics. 2010 Apr 15;26(8):1129-30.

Dittrich MT, Klau GW, Rosenwald A., Dandekar T and Muller T. *Identifying functional modules in protein-protein interaction networks: an integrated exact approach.* Bioinformatics 2008 24(13): i223-i231.

See Also

`analyzeGeneSetCollections`, `networkAnalysis`

Examples

```
## Not run:
library(org.Dm.eg.db)
library(KEGG.db)
##load data for enrichment analyses
data("KcViab_Data4Enrich")
```

```
##select hits
hits <- names(KcViab_Data4Enrich)[which(abs(KcViab_Data4Enrich) > 2)]
##set up a list of gene set collections
PW_KEGG <- KeggGeneSets(species = "Dm")
gscList <- list(PW_KEGG = PW_KEGG)
##create an object of class 'GSCA'
gsca <- new("GSCA", listOfGeneSetCollections=gscList,
geneList = KcViab_Data4Enrich, hits = hits)
##print gsca
gsca
##do preprocessing (KcViab_Data4Enrich has already been preprocessed)
gsca <- preprocess(gsca, species="Dm", initialIDs = "Entrez.gene",
keepMultipleMappings = TRUE, duplicateRemoverMethod = "max",
orderAbsValue = FALSE)
##print gsca again
gsca
##do hypergeometric tests and GSEA
gsca <- analyze(gsca, para = list(pValueCutoff = 0.05, pAdjustMethod
= "BH", nPermutations = 1000, minGeneSetSize = 100,exponent = 1))
##updated object
gsca
summarize(gsca)

## End(Not run)
```

`annotationConvertor` *Convert between different types of gene identifiers*

Description

This function converts an initial named data vector to the same vector but with a different identifier category, and removes the genes for which no mapping were found. This function can also take a matrix, with gene identifiers as row names.

Usage

```
annotationConvertor(geneList, species="Dm", initialIDs="Entrez.gene",
finalIDs="Entrez.gene", keepMultipleMappings=TRUE, verbose=TRUE)
```

Arguments

`geneList`	a named integer or numeric vector, or a matrix with rows named by gene identifiers
`species`	a single character value specifying the species for which the data should be read. The current version supports one of the following species: "Dm" ("Drosophila_melanogaster"), "Hs" ("Homo_sapiens"), "Rn" ("Rattus_norvegicus"), "Mm" ("Mus_musculus") and "Ce" ("Caenorhabditis_elegans").
`initialIDs`	a single character value specifying the type of initial identifiers for input 'geneList'. Current version can take one of the following types: "Ensembl.transcript", "Ensembl.prot", "Ensembl.gene", "Entrez.gene", "RefSeq", "Symbol" and "GenBank" for all supported species; "Flybase", "FlybaseCG" and "FlybaseProt" in addition for Drosophila Melanogaster; "wormbase" in addition for Caenorhabditis Elegans.

`finalIDs` a single character value specifying the type of initial identifiers for input 'geneList'. Current version can take one of the following types: "Ensembl.transcript", "Ensembl.prot", "Ensembl.gene", "Entrez.gene", "RefSeq", "Symbol" and "GenBank" for all supported species; "Flybase", "FlybaseCG" and "FlybaseProt" in addition for Drosophila Melanogaster; "wormbase" in addition for Caenorhabditis Elegans.

`keepMultipleMappings` a single logical value. If TRUE, the function keeps the entries with multiple mappings (first mapping is kept). If FALSE, the entries with multiple mappings will be discarded.

`verbose` a single logical value specifying to display detailed messages (when verbose=TRUE) or not (when verbose=FALSE)

Details

The entries that could not be mapped to any identifiers are removed from the resulting data vector/matrix. This function relies on the org.Dm.eg.dbpackage and therefore only maps

- from any identifier to an Entrez gene id, or
- from an Entrez gene ID to any identifier

Value

the same data vector/matrix but with names/row names converted.

Author(s)

Xin Wang, Camille Terfve

See Also

`mammalAnnotationConvertor`, `celAnnotationConvertor`, `drosoAnnotationConvertor`

Examples

```
library(org.Dm.eg.db)
##example 1: convert a named vector
x<-runif(10)
names(x)<-names(as.list(org.Dm.egSYMBOL2EG))[1:10]
xEntrez<-annotationConvertor(geneList=x, species="Dm", initialIDs="Symbol",
finalIDs="Entrez.gene")
##example 2: convert a data matrix with row names as gene ids
x<-cbind(runif(10),runif(10))
rownames(x)<-names(as.list(org.Dm.egSYMBOL2EG))[1:10]
xEntrez<-annotationConvertor(geneList=x, species="Dm", initialIDs="Symbol",
finalIDs="Entrez.gene")
```

appendGSTerms *Append gene set terms to GSCA results*

Description

This is a generic function.

When implemented as the S4 method for objects of class GSCA, this function finds corresponding annotation terms for KEGG and GO gene sets and inserts a column named "Gene.Set.Term" to each data frame in the GSCA results.

To use this function for objects of class GSCA:

appendGSTerms(object, keggGSCs=NULL, goGSCs=NULL)

Usage

```
appendGSTerms(object, ...)
```

Arguments

object	an object. When this function is implemented as the S4 method of class 'GSCA', this argument is an object of class 'GSCA'.
...	other arguments depending on class (see below for the arguments supported by the method of class 'GSCA')
	keggGSCs: a character vector of names of all KEGG gene set collections
	goGSCs: a character vector of names of all GO gene set collections

Details

This function makes the GSCA results more readable by appending a column of terms for KEGG and GO gene sets. To do this, the user needs to specify the names of the gene set collections based on KEGG and GO, respectively.

For each KEGG gene set, the species code in the KEGG id will be trimmed off, and then mapped to its corresponding annotation term using the function mget of the package AnnotationDbi.

For each GO gene set, the GO id will be mapped to corresponding GO term by the function Term of the package GO.db.

Value

In the end, this function will return an updated object of class GSCA.

Author(s)

Xin Wang <xw264@cam.ac.uk>

Examples

```
## Not run:
library(org.Dm.eg.db)
library(KEGG.db)
##load data for enrichment analyses
data("KcViab_Data4Enrich")
##select hits
hits <- names(KcViab_Data4Enrich)[which(abs(KcViab_Data4Enrich) > 2)]
##set up a list of gene set collections
PW_KEGG <- KeggGeneSets(species = "Dm")
gscList <- list(PW_KEGG = PW_KEGG)
##create an object of class 'GSCA'
gsca <- new("GSCA", listOfGeneSetCollections=gscList,
geneList = KcViab_Data4Enrich, hits = hits)
##print gsca
gsca
##do preprocessing (KcViab_Data4Enrich has already been preprocessed)
gsca <- preprocess(gsca, species="Dm", initialIDs = "Entrez.gene",
keepMultipleMappings = TRUE, duplicateRemoverMethod = "max",
orderAbsValue = FALSE)
##print gsca again
gsca
##do hypergeometric tests and GSEA
gsca <- analyze(gsca, para = list(pValueCutoff = 0.05, pAdjustMethod
= "BH", nPermutations = 1000, minGeneSetSize = 100,exponent = 1))
##append Kegg and GO gene set terms
gsca<-appendGSTerms(gsca, keggGSCs="PW_KEGG")

## End(Not run)
```

---

biogridDataDownload

Download and extract a network interaction matrix from a BioGRID data set

---

Description

This function downloads an interaction data set from the BioGRID into an user-specified folder and extracts an interaction matrix for a given species.

Usage

```
biogridDataDownload(link, species = "Dm", dataDirectory=".", verbose=TRUE)
```

Arguments

link	the link (url) from where the data should be downloaded (in tab2 format). If this argument is missing or NULL, the default link (version 3.1.71, valid on Dec. 5 2010) will be used.
species	a single character value specifying the species for which the data should be read. The current version supports one of the following species: "Dm" ("Drosophila_melanogaster"), "Hs" ("Homo_sapiens"), "Rn" ("Rattus_norvegicus"), "Mm" ("Mus_musculus"), "Ce" ("Caenorhabditis_elegans").

`dataDirectory`
: the directory to store downloaded file

`verbose`
: a single logical value specifying to display detailed messages (when verbose=TRUE) or not (when verbose=FALSE)

Details

This function is made to work on the tab2 format from the Biogrid (i.e. the first line is a header containing the columns names).

Value

a matrix with one row for each interaction, and three columns: InteractorA, InteractorB (both given by their Entrez Identifiers) and InteractionType (physical or genetic).

Author(s)

Camille Terfve, Xin Wang

References

Stark et al. *BioGRID: a general repository for interaction datasets.* Nucleic Acids Research 2006 34(Database Issue):D535-D539

See Also

networkAnalysis, preprocess

Examples

```
## Not run:
InteractionsData<-biogridDataDownload(species="Dm", dataDirectory="TestDir",
verbose=TRUE)

## End(Not run)
```

---

celAnnotationConvertor

Convert between different types of gene identifiers for Caenorhabditis Elegans

---

Description

This function converts an initial named data vector to the same vector but with a different identifier category, and removes the genes for which no mapping were found. This function can also take a matrix, with gene identifiers as row names.

Usage

```
celAnnotationConvertor(geneList, initialIDs = "Entrez.gene", finalIDs =
"Entrez.gene", keepMultipleMappings = TRUE, verbose = TRUE)
```

Arguments

geneList	a named integer or numeric vector, or a matrix with rows named by gene identifiers
initialIDs	a single character value specifying the type of initial identifiers for input 'geneList'. The current version can take one of the following types: "Ensembl.transcript", "Ensembl.prot", "Ensembl.gene", "Entrez.gene", "RefSeq", "Symbol", "GenBank" and "wormbase"
finalIDs	a single character value specifying the type of final identifiers to which users want to convert. The current version can take one of the following types: "Ensembl.transcript", "Ensembl.prot", "Ensembl.gene", "Entrez.gene", "RefSeq", "Symbol", "GenBank" and "wormbase"
keepMultipleMappings	a single logical value. If TRUE, the function keeps the entries with multiple mappings (first mapping is kept). If FALSE, the entries with multiple mappings will be discarded.
verbose	a single logical value specifying to display detailed messages (when verbose=TRUE) or not (when verbose=FALSE)

Details

This function relies on the org.Ce.eg.db package and therefore only maps

- from any identifier to an Entrez gene id, or
- from an Entrez gene ID to any identifier

The entries that could not be mapped to any identifiers are removed from the resulting data vector/matrix.

Value

the same vector/matrix but with names/rownames corresponding to a different type of identifiers

Author(s)

Camille Terfve, Xin Wang

See Also

drosoAnnotationConvertor, mammalAnnotationConvertor, annotationConvertor

Examples

```
library(org.Ce.eg.db)
##example 1: convert a named vector
x <- runif(10)
names(x) <- names(as.list(org.Ce.egSYMBOL2EG))[1:10]
xEntrez <- celAnnotationConvertor(geneList=x, initialIDs="Symbol",
finalIDs="Entrez.gene")
##example 2: convert a data matrix with row names as gene ids
x <- cbind(runif(10), runif(10))
rownames(x) <- names(as.list(org.Ce.egSYMBOL2EG))[1:10]
xEntrez <- celAnnotationConvertor(geneList=x, initialIDs="Symbol",
finalIDs="Entrez.gene")
```

`cellHTS2OutputStatTests` *Perform statistical tests on a cellHTS object*

Description

This function takes a normalized, configured and annotated cellHTS object and performs statistical tests on it for the significance of a set of observations for each condition tested in a high-throughput screen.

Usage

```
cellHTS2OutputStatTests(cellHTSobject, annotationColumn = "GeneID",
controls = "neg", alternative = "two.sided", logged = FALSE, tests =
"T-test")
```

Arguments

`cellHTSobject`	an object of class `cellHTS`
`annotationColumn`	a single character value specifying the name of the column in the fData(cellHTSobject) data frame from which the feature identifiers will be extracted
`controls`	a single character value specifying the name of the controls to be used as a control population in the two-sample tests (this HAS to be corresponding to how these control wells have been annotated in the column "controlStatus" of the fData(cellHTSobject) data frame). If nothing is specified, the function will look for negative controls labelled "neg".
`alternative`	a single character value specifying the alternative hypothesis: "two.sided", "less" or "greater"
`logged`	a single logical value specifying whether or not the data has been logged during the normalization process
`tests`	a single character value specifying the tests to be performed: "T-test", "MannWhitney" or "RankProduct". If nothing is specified, all three tests will be performed. Be aware that the Rank Product test is slower than the other two, and returns a percent false discovery (equivalent to a FDR, not a p-value).

Details

The tests are computed taking into account only the wells labelled "sample" in the column "controlStatus" of the fData(cellHTSobject).

The two sample tests compare the set of observations for one construct to the values obtained for a population considered as "control". The one-sample tests compare the set of observations for one construct to the median of all values obtained across all constructs labelled as "sample". This type of test assumes that most constructs are expected to show a negligible effect. It is therefore not advised to use this type of tests when the constructs tested have been pre-screened for being associated with a phenotype.

Please be aware that both types of tests are less reliable when the number of replicates for each construct is low.

Value

a matrix with two columns, one for each type of test (two-sample and one-sample test) except the Rank Product (no alternative), and a row for each construct (row names corresponding to the identifiers given by the "annotationcolumn" entry).

Author(s)

Camille Terfve, Xin Wang

References

Michael Boutros, Ligia P. Bras L and Wolfgang Huber. *Analysis of cell-based RNAi screens.* Genome Biology 7:7 R66 (2006)."

Examples

```
## Not run:
library(cellHTS2)
##load normalized cellHTS object (see the vignette for details about the
##preprocessing of this data set)
data("KcViab_Norm")
stats <- cellHTS2OutputStatTests(cellHTSobject=KcViab_Norm, alternative=
"two.sided", tests="T-test")

## End(Not run)
```

---

collectionGsea *Compute observed and permutation-based enrichment scores for a collection (list) of gene sets*

Description

This function computes observed and permutation-based scores associated with a gene set enrichment analysis for a collection of gene sets.

Usage

```
collectionGsea(collectionOfGeneSets, geneList, exponent=1, nPermutations=
1000, minGeneSetSize=15, verbose=TRUE)
```

Arguments

`collectionOfGeneSets`	a list of gene sets. Each gene set in the list is a character vector of gene identifiers.
`geneList`	a numeric or integer vector which has been named and ordered. It cannot contain any duplicates nor NAs.
`exponent`	a single numeric or integer value (set as 1 by default) specifying the exponent of the GSEA method.
`nPermutations`	a single numeric or integer value specifying the number of permutation tests for each gene set

`minGeneSetSize`
: a single numeric or integer value specifying the minimum size required for a gene set to be considered.

`verbose`
: a single logical value specifying to display detailed messages (when verbose=TRUE) or not (when verbose=FALSE)

Value

`Observed.scores`
: The observed scores for the given gene sets (a named vector)

`Permutation.scores`
: The scores for the permutation tests (one column for each permutation and a row for each gene set)

Author(s)

Camille Terfve, Xin Wang

References

Subramanian, A., Tamayo, P., Mootha, V. K., Mukherjee, S., Ebert, B. L., Gillette, M. A., Paulovich, A., Pomeroy, S. L., Golub, T. R., Lander, E. S. & Mesirov, J. P. (2005) *Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles.* Proc. Natl. Acad. Sci. USA 102, 15545-15550.

See Also

`FDRcollectionGsea`

Examples

```
##example 1
gl <- runif(100, min=0, max=5)
gl <- gl[order(gl, decreasing=TRUE)]
names(gl) <- as.character(sample(x=seq(from=1, to=100, by=1), size=100,
replace=FALSE))
gs1 <- sample(names(gl), size=20, replace=FALSE)
gs2 <- sample(names(gl), size=20, replace=FALSE)
gsc <- list(subset1=gs1, subset2=gs2)
GSCscores <- collectionGsea(collectionOfGeneSets=gsc, geneList=gl,
exponent=1, nPermutations=1000, minGeneSetSize=5)
GSCpvalues <- permutationPvalueCollectionGsea(permScores=
GSCscores$Permutation.scores, dataScores=GSCscores$Observed.scores)
##example 2
## Not run:
library(org.Dm.eg.db)
library(KEGG.db)
##load phenotype vector (see the vignette for details about the
##preprocessing of this data set)
data("KcViab_Data4Enrich")
DM_KEGG <- KeggGeneSets(species="Dm")
GSCscores <- collectionGsea(collectionOfGeneSets=DM_KEGG, geneList=
KcViab_Data4Enrich, exponent=1, nPermutations=1000, minGeneSetSize=100)
GSCpvalues <- permutationPvalueCollectionGsea(permScores=
GSCscores$Permutation.scores, dataScores=GSCscores$Observed.scores)
```

```
## End(Not run)
```

`data-KcViab` *A Sample data set of the package HTSanalyzeR*

Description

'KcViab_Norm' is a normalized cellHTS object (KcViab_Norm) generated from a genome-wide RNAi screen of cell viability in *Drosophila* Kc167 cells (see help(KcViab)).

'KcViab_Data4Enrich' is a vector of phenotypes summarized and preprocessed from 'KcViab_Norm' (see section 4.1 of the vignette of package *HTSanalyzeR* for details).

'KcViab_PVals' is a vector of p-values generated by statistical tests for the summarized data of 'KcViab_Norm' (see section 5.1 of the vignetted of package *HTSanalyzeR* for details).

'KcViab_GSCA' is an object of class 'GSCA' (Gene Set Collection Analysis) (see section 4.2 of the vignette of package *HTSanalyzeR* for details).

'KcViab_NWA' is an object of class 'NWA' (NetWork Analysis) (see section 5.2 of the vignetted of package *HTSanalyzeR* for details).

'Biogrid_DM_Interactome' is an object of class *graphNEL* built from the interaction data set for *Drosophila Melanogaster* downloaded from the *BioGRID* database (version 3.1.71, accessed on Dec. 5, 2010).

Usage

```
##see examples for details
```

References

Boutros, M., Kiger, A.A., Armknecht,S., Kerr,K., Hild,M., Koch,B., Haas, S.A., Heidelberg Fly Array Consortium, Paro,R. and Perrimon, N. (2004) Genome-wide RNAi analysis of growth and viability in Drosophila cells, *Science* **303**:832–5.

Examples

```
data("KcViab_Norm")
data("KcViab_Data4Enrich")
data("KcViab_PVals")
data("KcViab_GSCA")
data("KcViab_NWA")
data("Biogrid_DM_Interactome")
data("Biogrid_DM_Mat")
```

`drosoAnnotationConvertor` *Convert between different types of gene identifiers for Drosophila Melanogaster*

Description

This function converts an initial named data vector to the same vector but with a different identifier category, and removes the genes for which no mapping were found. This function can also take a matrix, with gene identifiers as row names.

Usage

```
drosoAnnotationConvertor(geneList, initialIDs = "Entrez.gene", finalIDs =
"Entrez.gene", keepMultipleMappings = TRUE, verbose=TRUE)
```

Arguments

`geneList`	a named integer or numeric vector, or a matrix with rows named by gene identifiers
`initialIDs`	a single character value specifying the type of initial identifiers for input 'geneList'. The current version can take one of the following types: "Ensembl.transcript", "Ensembl.prot", "Ensembl.gene", "Entrez.gene", "RefSeq", "Symbol", "GenBank", "Flybase", "FlybaseCG", "FlybaseProt".
`finalIDs`	a single character value specifying the type of final identifiers to which users want to convert. The current version can take one of the following types: "Ensembl.transcript", "Ensembl.prot", "Ensembl.gene", "Entrez.gene", "RefSeq", "Symbol", "GenBank", "Flybase", "FlybaseCG", "FlybaseProt".
`keepMultipleMappings`	a single logical value. If TRUE, the function keeps the entries with multiple mappings (first mapping is kept). If FALSE, the entries with multiple mappings will be discarded.
`verbose`	a single logical value specifying to display detailed messages (when verbose=TRUE) or not (when verbose=FALSE)

Details

The entries that could not be mapped to any identifiers are removed from the resulting data vector/matrix. This function relies on the org.Dm.eg.dbpackage and therefore only maps

- from any identifier to an Entrez gene id, or
- from an Entrez gene ID to any identifier

Value

the same data vector/matrix but with names/row names converted.

Author(s)

Camille Terfve, Xin Wang

See Also

mammalAnnotationConvertor, celAnnotationConvertor, annotationConvertor

Examples

```
library(org.Dm.eg.db)
##example 1: convert a named vector
x <- runif(10)
names(x) <- names(as.list(org.Dm.egSYMBOL2EG))[1:10]
xEntrez <- drosoAnnotationConvertor(geneList=x, initialIDs="Symbol",
finalIDs="Entrez.gene")
##example 2: convert a data matrix with row names as gene ids
x <- cbind(runif(10), runif(10))
rownames(x) <- names(as.list(org.Dm.egSYMBOL2EG))[1:10]
xEntrez <- drosoAnnotationConvertor(geneList=x, initialIDs="Symbol",
finalIDs="Entrez.gene")
```

---

duplicateRemover *Remove duplicates in a named vector of phenotypes*

Description

This function gets rid of the duplicates in a vector of phenotypes with gene identifiers as names. It is used to prepare the named vector of phenotypes for the over-representation and enrichment analysis.

Usage

```
duplicateRemover(geneList, method = "max")
```

Arguments

geneList	a single named numeric or integer vector with gene identifiers as names
method	a single character value specifying the method to remove the duplicates (should the minimum, maximum or average observation for a same construct be kept). The current version provides "min" (minimum), "max" (maximum), "average" and "fc.avg" (fold change average). The minimum and maximum should be understood in terms of absolute values (i.e. min/max effect, no matter the sign). The fold change average method converts the fold changes to ratios, averages them and converts the average back to a fold change.

Value

a named vector of phenotypes with duplicates removed

Author(s)

Camille Terfve, John C. Rose and Xin Wang

See Also

preprocess

Examples

```
x<-c(5,1,3,-2,6)
names(x)<-c("gene1","gene3","gene7","gene3","gene4")
xprocessed<-duplicateRemover(geneList=x,method="max")
```

`FDRcollectionGsea` *Compute the GSEA false discovery rates for a collection (list) of gene sets*

Description

This function computes the GSEA fdr over a list of gene sets

Usage

```
FDRcollectionGsea(permScores, dataScores)
```

Arguments

`permScores`	a numeric matrix of permutation-based scores resulting from the output of collectionGsea
`dataScores`	a named numeric vector of observed scores resulting from the output of collectionGsea

Value

a named numeric vector of FDR, one for each gene set

Author(s)

Camille Terfve, Xin Wang

References

Subramanian, A., Tamayo, P., Mootha, V. K., Mukherjee, S., Ebert, B. L., Gillette, M. A., Paulovich, A., Pomeroy, S. L., Golub, T. R., Lander, E. S. & Mesirov, J. P. (2005) *Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles.* Proc. Natl. Acad. Sci. USA 102, 15545-15550.

See Also

`collectionGsea`, `permutationPvalueCollectionGsea`

Examples

```
##example 1
gl <- runif(100, min=0, max=5)
gl <- gl[order(gl, decreasing=TRUE)]
names(gl) <- as.character(sample(x=seq(from=1, to=100, by=1), size=100,
replace=FALSE))
gs1 <- sample(names(gl), size=20, replace=FALSE)
gs2 <- sample(names(gl), size=20, replace=FALSE)
```

```
gscs <- list(gs1=gs1, gs2=gs2)
GSCscores <- collectionGsea(collectionOfGeneSets=gscs, geneList=gl,
exponent=1, nPermutation=1000, minGeneSetSize=5)
GSCfdrs <- FDRcollectionGsea(permScores=GSCscores$Permutation.scores,
dataScores=GSCscores$Observed.scores)
##example 2 (see the vignette for details about the preprocessing of this
##data set)
## Not run:
library(org.Dm.eg.db)
library(KEGG.db)
data("KcViab_Data4Enrich")
DM_KEGG <- KeggGeneSets(species="Dm")
GSCscores <- collectionGsea(collectionOfGeneSets=DM_KEGG, geneList=
KcViab_Data4Enrich, exponent=1, nPermutations=1000, minGeneSetSize=100)
GSCfdrs <- FDRcollectionGsea(permScores=GSCscores$Permutation.scores,
dataScores=GSCscores$Observed.scores)

## End(Not run)
```

`getTopGeneSets` *Select top significant gene sets from GSEA results*

Description

This is a generic function.

When implemented as the S4 method of class GSCA, this function selects top significant gene sets from GSEA results for user-specified gene collections. If 'ntop' is given, then top 'ntop' significant gene sets in gene set collections 'gscs' will be selected and their names will be returned. If 'allSig=TRUE', then all significant (adjusted p-value < 'pValueCutoff' see help("analyze")) gene sets will be selected and their names will be returned.

To use this function for objects of class GSCA:

getTopGeneSets(object, resultName, gscs, ntop=NULL, allSig=FALSE)

Usage

```
getTopGeneSets(object, ...)
```

Arguments

`object`	an object. When this function is implemented as the S4 method of class GSCA, this argument is an object of class GSCA.
`...`	other arguments (see below for the arguments supported by the method of class GSCA)

resultName: a single character value: 'HyperGeo.results' or 'GSEA.results'

gscs: a character vector specifying the names of gene set collections from which the top significant gene sets will be selected

ntop: a single integer or numeric value specifying to select how many gene sets of top significance.

allSig: a single logical value. If 'TRUE', all significant gene sets (GSEA adjusted p-value < 'pValueCutoff' of slot 'para') will be selected; otherwise, only top 'ntop' gene sets will be selected.

Value

a list of character vectors, each of which contains the names of top significant gene sets for each gene set collection

Author(s)

Xin Wang <xw264@cam.ac.uk>

Examples

```
## Not run:
library(org.Dm.eg.db)
library(KEGG.db)
##load data for enrichment analyses
data("KcViab_Data4Enrich")
##select hits
hits <- names(KcViab_Data4Enrich)[which(abs(KcViab_Data4Enrich) > 2)]
##set up a list of gene set collections
PW_KEGG <- KeggGeneSets(species = "Dm")
gscList <- list(PW_KEGG = PW_KEGG)
##create an object of class 'GSCA'
gsca <- new("GSCA", listOfGeneSetCollections=gscList, geneList =
KcViab_Data4Enrich, hits = hits)
##print summary of gsca
summarize(gsca)
##do preprocessing (KcViab_Data4Enrich has already been preprocessed)
gsca <- preprocess(gsca, species="Dm", initialIDs = "Entrez.gene",
keepMultipleMappings = TRUE, duplicateRemoverMethod = "max",
orderAbsValue = FALSE)
##print summary of gsca again
summarize(gsca)
##do hypergeometric tests and GSEA
gsca <- analyze(gsca, para = list(pValueCutoff = 0.05, pAdjustMethod
= "BH", nPermutations = 1000, minGeneSetSize = 100,exponent = 1))
##print summary of results
summarize(gsca, what="Result")
##print top significant gene sets in GO.BP
topPWKEGG<-getTopGeneSets(gsca, "GSEA.results", "PW_KEGG", allSig=TRUE)

## End(Not run)
```

GOGeneSets *Create a list of gene sets based on GO terms*

Description

This function creates a list of gene sets based on GO terms. It is species-specific, and returns a list of gene sets, each of which is a character vector of Entrez identifiers.

Usage

```
GOGeneSets(species = "Dm", ontologies = "MF")
```

Arguments

species	a single character value specifying a choice of species: "Dm" ("Drosophila_melanogaster"), "Hs" ("Homo_sapiens"), "Rn" ("Rattus_norvegicus"), "Mm" ("Mus_musculus") or "Ce" ("Caenorhabditis_elegans"))
ontologies	a single character value or a character vector specifying an ontology or multiple ontologies. The current version provides the following choices: "BP", "CC" and "MF"

Details

This function relies on the following packages: GSEABase, GO.db, and either org.Hs.eg.db, org.Mm.eg.db, org.Rn.eg.db, org.Ce.eg.db, org.Dm.eg.db.

Value

a list of gene sets, with names as GO IDs. Each gene set is a character vector of Entrez identifiers.

Author(s)

Camille Terfve

See Also

KeggGeneSets

Examples

```
library(GO.db)
library(org.Dm.eg.db)
Dm_GO_CC<-GOGeneSets(species="Dm",ontologies=c("CC"))
```

GSCA-class *An S4 class for Gene Set Collection Analyses on high-throughput screens*

Description

This S4 class includes a series of methods to do gene set enrichment analysis and hypergeometric tests for high-throughput screens.

Objects from the Class

Objects of class GSCA can be created from `new("GSCA", listOfGeneSetCollections, geneList, hits)` (see the examples below)

Slots

`listOfGeneSetCollections`: a list of gene set collections (a 'gene set collection' is a list of gene sets).

`geneList`: a numeric or integer vector of phenotypes named by gene identifiers.

`hits`: a character vector of the gene identifiers (used as hits in the hypergeometric tests).

`para`: a list of parameters for hypergeometric tests and GSEA. These parameters are `pValueCutoff`, `pAdjustMethod`, `nPermutations`, `minGeneSetSize` and `exponent` (see function `analyzeGeneSetCollections` for detailed descriptions about these parameters).

`result`: a list of results (see the returned values in the function `analyzeGeneSetCollections`).

`summary`: a list of summary information for `listOfGeneSetCollections`, `geneList`, `hits`, `para`, and `result`.

`preprocessed`: a single logical value specifying whether or not the input data has been preprocessed.

Methods

An overview of methods with class-specific functionality: More detailed introduction can be found in help for each specific function.

`preprocess` do preprocessing on input vectors of phenotypes and hits including: a) removing NAs in the `geneList` and `hits`; b) invoking function `duplicateRemover` to process duplicated phenotypes (see `duplicateRemover` for more details); c) invoking function `annotationConvertor` to convert annotations; d) ranking phenotypes in a decreasing order.

`analyze` perform hypergeometric tests and Gene Set Enrichment Analysis based on input parameter list `para`.

`appendGSTerms` append gene set terms to GSCA results

`summarize` print summary information about `listOfGeneSetCollections`, `geneList`, `hits`, `para`, and `result`.

`getTopGeneSets` select top significant gene sets from `object@results$`` `resultName` `` by setting `ntop` or `allSig`.

`writeHits` write observed hits in gene sets for hypergeometric tests.

`viewGSEA` view a figure of GSEA results for a gene set in a gene set collection.

`plotGSEA` plot and save figures of GSEA results for top significant gene sets in a gene set collection.

`viewEnrichMap` plot an enrichment map for GSEA or hypergeometric test results

`plotEnrichMap` plot and save an enrichment map for GSEA or hypergeometric test results

`report` generate html reports.

Author(s)

Xin Wang <xw264@cam.ac.uk>

See Also

`preprocess` `analyze` `appendGSTerms` `summarize` `getTopGeneSets` `writeHits` `viewGSEA` `plotGSEA` `viewEnrichMap` `plotEnrichMap` `report`

Examples

```
## Not run:
library(org.Dm.eg.db)
library(KEGG.db)
library(AnnotationDbi)
library(igraph)
##load data for enrichment analyses
data("KcViab_Data4Enrich")
##select hits
hits <- names(KcViab_Data4Enrich)[which(abs(KcViab_Data4Enrich) > 2)]
##set up a list of gene set collections
PW_KEGG <- KeggGeneSets(species = "Dm")
gscs <- list(PW_KEGG = PW_KEGG)
##create an object of class 'GSCA'
gsca <- new("GSCA", listOfGeneSetCollections=gscs, geneList =
KcViab_Data4Enrich, hits = hits)
##do preprocessing (KcViab_Data4Enrich has already been preprocessed)
gsca <- preprocess(gsca, species="Dm", initialIDs = "Entrez.gene",
keepMultipleMappings = TRUE, duplicateRemoverMethod = "max",
orderAbsValue = FALSE)
##do hypergeometric tests and GSEA
gsca <- analyze(gsca, para = list(pValueCutoff = 0.05, pAdjustMethod
= "BH", nPermutations = 1000, minGeneSetSize = 60, exponent = 1))
##print summary information
summarize(gsca)
##get all significant gene sets in "PW_KEGG"
sigGSs<-getTopGeneSets(gsca, "GSEA.results", "PW_KEGG", allSig=TRUE)
##view a GSEA figure
viewGSEA(gsca, gscName="PW_KEGG", gsName=sigGSs[["PW_KEGG"]][1])
dev.off()
##append gene set terms to results
gsca<-appendGSTerms(gsca, keggGSCs="PW_KEGG")
##view an enrichment map for GSEA results
eb<-viewEnrichMap(gsca, gscs="PW_KEGG", allSig=TRUE, gsNameType="term",
displayEdgeLabel=FALSE, layout="layout.fruchterman.reingold")
##write html reports
report(object = gsca, experimentName = "GSCATest", species = "Dm",
allSig = TRUE, keggGSCs = "PW_KEGG", reportDir="GSCATestReport")
##browse the index page
browseURL(file.path(getwd(), "GSCATestReport", "index.html"))

## End(Not run)
```

`gseaPlots` *Plot GSEA results for one gene set*

Description

This function takes in the output of gseaScores and the named vector of phenotypes, and plots the positions of genes of the gene set in the ranked phenotype vector and the location of the enrichment score.

Usage

```
gseaPlots(runningScore, enrichmentScore, positions, geneList)
```

Arguments

runningScore	a single numeric value specifying the enrichment score returned from the function "gseaScores"
enrichmentScore	a numeric vector of running sum scores (only in mode "graph")
positions	a numeric vector specifying positions in the ranked phenotype vector of the genes in the gene set (only in mode "graph")
geneList	a numeric or integer vector of phenotypes in descending or ascending order with elements named by their EntrezIds (no duplicates nor NA values)

Author(s)

Camille Terfve, Xin Wang

References

Subramanian, A., Tamayo, P., Mootha, V. K., Mukherjee, S., Ebert, B. L., Gillette, M. A., Paulovich, A., Pomeroy, S. L., Golub, T. R., Lander, E. S. & Mesirov, J. P. (2005) *Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles.* Proc. Natl. Acad. Sci. USA 102, 15545-15550.

See Also

plotGSEA, viewGSEA

Examples

```
## Not run:
library(org.Dm.eg.db)
library(KEGG.db)
##load phenotype vector (see the vignette for details about the
##preprocessing of this data set)
data("KcViab_Data4Enrich")
DM_KEGG <- KeggGeneSets(species="Dm")
GSCscores <- gseaScores(geneList=KcViab_Data4Enrich, geneSet=DM_KEGG[[1]],
exponent=1, mode="graph")
gseaPlots(runningScore=GSCscores$runningScore, enrichmentScore=
GSCscores$enrichmentScore, positions=GSCscores$positions,
geneList=KcViab_Data4Enrich)

## End(Not run)
```

---

gseaScores *Compute enrichment scores for GSEA (Gene Set Enrichment Analysis)*

Description

'gseaScores' computes the enrichment score, running sum scores and positions of hits for GSEA on one gene set.

'gseaScoresBatch' computes enrichment scores for both input 'geneList' and its permutations for GSEA on one gene set.

'gseaScoresBatchParallel' computes enrichment scores for both input 'geneList' and their permutations for GSEA on multiple gene sets in parallel.

Usage

```
gseaScores(geneList, geneSet, exponent=1, mode = "score")

gseaScoresBatch(geneList, geneNames.perm, geneSet, exponent=1,
nPermutations=1000)

gseaScoresBatchParallel(geneList, geneNames.perm, collectionOfGeneSets,
exponent=1, nPermutations=1000)
```

Arguments

geneList	a numeric or integer vector of phenotypes in descending or ascending order with elements named by their gene identifiers (no duplicates nor NA values)
geneNames.perm	a character matrix including permuted gene identifiers of input 'geneList'. The first column of this matrix should be the gene identifiers of 'geneList', while other columns are vectors of permuted gene identifiers.
geneSet	a character vector specifying a gene set (no names, just a vector of characters corresponding to the IDs)
collectionOfGeneSets	a list of gene sets. Each gene set in the list is a character vector of gene identifiers
exponent	a single integer or numeric value used in weighting phenotypes in GSEA
nPermutations	a single integer or numeric value specifying the number of permutations for deriving p-values in GSEA
mode	a single character value specifying to return only a score (if set as "score"), or all the necessary elements to make a plot (if "graph") (see "gseaPlot" for more details).

Details

The type of identifiers used in the gene sets and gene list must obviously match.

Value

* gseaScores will return:

enrichmentScore	a single numeric value of the enrichment score
runningScore	a numeric vector of running sum scores (only in mode "graph")
positions	a numeric vector of positions in the ranked phenotype vector of the genes in the gene set (only in mode "graph")

* gseaScoresBatch will return a list consisting of:

scoresObserved	a single numeric value of the enrichment score for input 'geneList'
scoresperm	a numeric vector of enrichment scores for permutation tests

* gseaScoresBatchParallel will return a matrix, in which one column is the 'gseaScoresBatch' result for each gene set.

Author(s)

Xin Wang, Camille Terfve

References

Subramanian, A., Tamayo, P., Mootha, V. K., Mukherjee, S., Ebert, B. L., Gillette, M. A., Paulovich, A., Pomeroy, S. L., Golub, T. R., Lander, E. S. & Mesirov, J. P. (2005) *Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles.* Proc. Natl. Acad. Sci. USA 102, 15545-15550.

Examples

```
##example 1
gl <- runif(100, min=0, max=5)
gl <- gl[order(gl, decreasing=TRUE)]
names(gl) <- as.character(sample(x=seq(from=1, to=100, by=1), size=100,
replace=FALSE))
gs <- sample(names(gl), size=20, replace=FALSE)
##GSEA for input phenotype vector on a single gene set
gsea <- gseaScores(geneList=gl, geneSet=gs, mode="score", exponent=1)
##GSEA for both input phenotype vector and permutation tests on a single gene set
nPermutations <- 100
glPerm <- sapply(1:nPermutations, function(n) names(gl)[sample(1:length(gl),
length(gl), replace=FALSE)])
glPerm <- cbind(names(gl), glPerm)
gseaBatch<-gseaScoresBatch(geneList=gl, geneNames.perm=glPerm, geneSet=gs,
nPermutations=100, exponent=1)
##example 2
## Not run:
library(KEGG.db)
library(org.Dm.eg.db)
library(snow)
##load phenotype vector (see the vignette for details about the
##preprocessing of this data set)
data("KcViab_Data4Enrich")
DM_KEGG <- KeggGeneSets(species="Dm")
##GSEA for input gene list on a single gene set
test <- gseaScores(geneList=KcViab_Data4Enrich, geneSet=DM_KEGG[[1]],
exponent=1, mode="graph")
##GSEA for both input gene list and permutation tests on multiple gene
##sets in parallel
nPermutations <- 100
glPerm <- sapply(1:nPermutations, function(n) names(KcViab_Data4Enrich)[
sample(1:length(KcViab_Data4Enrich), length(KcViab_Data4Enrich),
replace=FALSE)])
glPerm<-cbind(names(KcViab_Data4Enrich), glPerm)
options(cluster=makeCluster(4,"SOCK"))
gseaBatchPar <- gseaScoresBatchParallel(geneList=KcViab_Data4Enrich,
geneNames.perm=glPerm, collectionOfGeneSets=DM_KEGG[1:10], nPermutations=100,
exponent=1)
if(is(getOption("cluster"),"cluster")) {
stopCluster(getOption("cluster"))
options(cluster=NULL)
}

## End(Not run)
```

`HTSanalyzeR4cellHTS2` *An analysis pipeline for cellHTS2 objects*

Description

This function writes an html report following a complete analyses of a dataset based on the two classes `GSCA` (Gene Set Collection Analysis) and `NWA` (NetWork Analysis) of this package.

Usage

```
HTSanalyzeR4cellHTS2(
normCellHTSobject,
scoreSign = "-",
scoreMethod = "zscore",
summarizeMethod = "mean",
annotationColumn = "GeneID",
species = "Dm",
initialIDs = "FlybaseCG",
duplicateRemoverMethod = "max",
orderAbsValue = FALSE,
listOfGeneSetCollections,
cutoffHitsEnrichment = 2,
pValueCutoff = 0.05,
pAdjustMethod = "BH",
nPermutations = 1000,
minGeneSetSize = 15,
exponent = 1,
keggGSCs,
goGSCs,
nwStatsControls = "neg",
nwStatsAlternative = "two.sided",
nwStatsTests = "T-test",
nwStatsColumns = c("t.test.pvalues.two.samples", "t.test.pvalues.one.sample"),
nwAnalysisFdr = 0.001,
nwAnalysisGenetic = FALSE,
interactionMatrix = NULL,
nwAnalysisOrder = 2,
ntop = NULL,
allSig = TRUE,
reportDir = "HTSanalyzerReport",
verbose = TRUE
)
```

Arguments

`normCellHTSobject`	a normalized, configured and annotated cellHTS object
`scoreSign`	a single character value specifying the 'sign' argument for the scoring function from cellHTS2 (see 'scoreReplicates')

`scoreMethod` a single character value specifying the 'method' argument for the scoring function from cellHTS2 (see 'scoreReplicates')

`summarizeMethod`
a summary argument for the summarization function from cellHTS2 (see 'summarizeReplicates')

`annotationColumn`
a single character value specifying the name of the column in the fData (cellHTSobject) data frame from which the feature identifiers will be extracted

`species` a single character value specifying the species for which the data should be read. The current version supports one of the following species: "Dm" ("Drosophila_melanogaster"), "Hs" ("Homo_sapiens"), "Rn" ("Rattus_norvegicus"), "Mm" ("Mus_musculus"), "Ce" ("Caenorhabditis_elegans").

`initialIDs` a single character value specifying the type of initial identifiers for input 'geneList'. Current version can take one of the following types: "Ensembl.transcript", "Ensembl.prot", "Ensembl.gene", "Entrez.gene", "RefSeq", "Symbol" and "GenBank" for all supported species; "Flybase", "FlybaseCG" and "FlybaseProt" in addition for Drosophila Melanogaster; "wormbase" in addition for Caenorhabditis Elegans.

`duplicateRemoverMethod`
a single character value specifying the method to remove the duplicates (should the minimum, maximum or average observation for a same construct be kept). The current version provides "min" (minimum), "max" (maximum), "average" and "fc.avg" (fold change average). The minimum and maximum should be understood in terms of absolute values (i.e. min/max effect, no matter the sign). The fold change average method converts the fold changes to ratios, averages them and converts the average back to a fold change.

`orderAbsValue`
a single logical value determining whether the values should be converted to absolute value and then ordered (if TRUE), or ordered as they are (if FALSE).

`listOfGeneSetCollections`
a list of gene set collections (a 'gene set collection' is a list of gene sets). Even if only one collection is being tested,it must be entered as an element of a 1-element list, e.g. `ListOfGeneSetCollections = list(YourOneGeneSetCollectior` Naming the elements of listOfGeneSetCollections will result in these names being associated with the relevant data frames in the output (meaningful names are advised)

`cutoffHitsEnrichment`
a single numeric or integer value specifying the cutoff that is used in the definition of the hits for the hypergeometric tests in the over- representation analysis. This cutoff is used in absolute value, since it is applied on scores, i.e. a cutoff of 2 when using z-scores means that we are selecting values that are two standard deviations away from the median of all samples. Therefore, the cutoff should be a positive number.

`pValueCutoff` a single numeric value specifying the cutoff for p-values considered significant

`pAdjustMethod`
a single character value specifying the p-value adjustment method to be used (see 'p.adjust' for details)

`nPermutations`
a single integer or numeric value specifying the number of permutations for deriving p-values in GSEA

`minGeneSetSize`
: a single integer or numeric value specifying the minimum number of elements in a gene set that must map to elements of the gene universe. Gene sets with fewer than this number are removed from both hypergeometric analysis and GSEA.

`exponent`
: a single integer or numeric value used in weighting phenotypes in GSEA (see "gseaScores" function)

`keggGSCs`
: a character vector of names of all KEGG gene set collections. This will help create web links for KEGG terms.

`goGSCs`
: a character vector of names of all GO gene set collections. This will help create web links for GO terms.

`nwStatsControls`
: a single character value specifying the name of the controls to be used as a control population in the two-sample tests (this HAS to be corresponding to how these control wells have been annotated in the column "controlStatus" of the fData(cellHTSobject) data frame). If nothing is specified, the function will look for negative controls labelled "neg".

`nwStatsAlternative`
: a single character value specifying the alternative hypothesis: "two.sided", "less" or "greater"

`nwStatsTests`
: a single character value specifying the tests to be performed: "T-test", "MannWhitney" or "RankProduct". If nothing is specified, all three tests will be performed. Be aware that the Rank Product test is slower than the other two, and returns a percent false discovery (equivalent to a FDR, not a p-value).

`nwStatsColumns`
: a character vector of any (relevant, i.e. that is produced in the tests) combination of "t.test.pvalues.two.samples", "t.test.pvalues.one.sample", "mannW.test.pvalues.one.sample", "mannW.test.pvalues.two.samples", "rank.product.pfp.greater", "rank.product.pfp.less"

`nwAnalysisFdr`
: a single numeric value specifying the FDR used in the networkAnalysis function for the scores calculation

`nwAnalysisGenetic`
: a single logical value indicating if the genetic interaction data of the Biogrid dataset were kept in the network analysis

`interactionMatrix`
: an interaction matrix including columns 'InteractionType', 'InteractorA' and 'InteractorB'. If this matrix is available, the interactome can be directly built based on it.

`nwAnalysisOrder`
: the order used in the networkAnalysis function for the scores calculation

`ntop`
: the number of plots to be produced for the GSEA analysis. For each gene set collection, plots are produced for the "nplots" most significant p-values.

`allSig`
: a single logical value indicating whether or not to generate plots for all significant gene sets. A gene set is significant if its corresponding adjusted p-value is less than the `pValueCutoff` set in function `analyze` (see function `analyze` for more details).

`reportDir`
: a single character value specifying the directory to store reports

`verbose`
: a single logical value indicating to display detailed messages (when verbose= TRUE) or not (when verbose=FALSE)

Details

This pipeline function performs gene set over-representation, enrichment analyses and identifies enriched subnetworks, writes a report and saves results as 'RData' in the user-specified directory.

For the KEGG and GO gene sets, this function produces links to the relevant database entries. To make the report more readable, the user can append a column named 'Gene.Set.Term' to each data frame in the result slot of the object of class GSCA using the method appendGSTerms. To ensure that the function works properly, the gene set ID should comply with the following requirements:

- for KEGG 3 letters (species ID) followed by a set of numbers and then any number of non-digit characters
- for GO "GO:" immediately followed by the GO term number (digits only) and any number of non-digits characters.

Value

Produce a set of html pages and save GSCA and NWA objects. The report starts from the page called "index.html" to navigate those pages.

Author(s)

Camille Terfve, Xin Wang

Examples

```
## Not run:
library(org.Dm.eg.db)
library(GO.db)
library(KEGG.db)
library(cellHTS2)
library(BioNet)
library(igraph)
#prepare data from cellHTS2
experimentName <- "KcViab"
dataPath <- system.file(experimentName, package = "cellHTS2")
x <- readPlateList("Platelist.txt", name = experimentName, path=dataPath,
verbose=TRUE)
x <- configure(x, descripFile = "Description.txt", confFile =
"Plateconf.txt",logFile = "Screenlog.txt", path = dataPath)
xn <- normalizePlates(x, scale = "multiplicative", log = FALSE, method =
"median", varianceAdjust = "none")
xn <- annotate(xn, geneIDFile = "GeneIDs_Dm_HFA_1.1.txt",path = dataPath)
cellHTS2DrosoData<-xn
#prepare a list of gene set collections
GO_MF <- GOGeneSets(species = "Dm",ontologies = c("MF"))
PW_KEGG <- KeggGeneSets(species = "Dm")
gscList <- list(GO_MF = GO_MF, PW_KEGG = PW_KEGG)
#note: set computer cluster here to do parallel computing
#options(cluster=makeCluster(4,"SOCK"))
HTSanalyzeR4cellHTS2(
normCellHTSobject=cellHTS2DrosoData,
annotationColumn="GeneID",
species=c("Dm"),
initialIDs="FlybaseCG",
listOfGeneSetCollections=gscList,
cutoffHitsEnrichment=2,
```

```
minGeneSetSize=200,
keggGSCs="PW_KEGG",
goGSCs=c("GO_MF"),
allSig=TRUE,
reportDir="HTSanalyzerReport",
verbose=TRUE
)
#note: stop the cluster if you created
#if(is(getOption("cluster"),"cluster")) {
# stopCluster(getOption("cluster"))
# options(cluster=NULL)
#}
##browse the index page
browseURL(file.path(getwd(), "HTSanalyzerReport", "index.html"))

## End(Not run)
```

`HTSanalyzeR-package`

HTSanalyzeR Package Overview

Description

This package provides classes and methods for gene set over-representation, enrichment and network analyses on high-throughput screens. The over- representation analysis is performed based on hypergeometric tests. The enrichment analysis is based on the GSEA algorithm (Subramanian et al. PNAS 2005). The network analysis identifies enriched subnetworks based on algorithms from the BioNet package (Beisser et al., Bioinformatics 2010). A pipeline is also specifically designed for cellHTS2 object to perform integrative network analyses of high-throughput RNA interference screens. The users can build their own analysis pipeline for their own data set based on this package.

Details

Package:	HTSAnalyzeR
Type:	Package
Version:	2.3.5
Date:	2011-03-03
License:	Artistic-2.0
LazyLoad:	yes

The most important classes in this package are 'GSCA' (Gene Set Collection Analyses) and 'NWA' (NetWork Analyses). As an example, a pipeline (see function 'HTSanalyzeR4cellHTS2') is developed in this package for cellHTS2 screen analyses. Based on these two classes and other functions, users can design their own pipelines specifically for their own data sets.

Full help on classes and associated functions is available from within class help pages.

Introductory information on the use of classes and pipeline are available in the vignette, type`openVignette()`.

A full listing of documented topics is available in HTML view by typing `help.start()` and selecting the `HTSanalyzeR` package from the `Packages` menu or via `library(help="HTSanalyzeR")`.

Author(s)

Xin Wang, Camille D.A. Terfve, John C. Rose, and Florian Markowetz Maintainer: Xin Wang <Xin.Wang@cancer.org.uk>

References

Subramanian, A., Tamayo, P., Mootha, V. K., Mukherjee, S., Ebert, B. L., Gillette, M. A., Paulovich, A., Pomeroy, S. L., Golub, T. R., Lander, E. S. & Mesirov, J. P. (2005) *Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles.* Proc. Natl. Acad. Sci. USA 102, 15545-15550.

Beisser D, Klau GW, Dandekar T, Muller T, Dittrich MT. BioNet: an R-Package for the functional analysis of biological networks. Bioinformatics. 2010 Apr 15;26(8):1129-30.

Dittrich MT, Klau GW, Rosenwald A., Dandekar T and Muller T. *Identifying functional modules in protein-protein interaction networks: an integrated exact approach.* Bioinformatics 2008 24(13):i223-i231.

---

hyperGeoTest *Performs hypergeometric tests for over-representation analysis*

Description

This function takes in a single gene set (geneSet), a vector of gene identifiers for all tested genes (universe), a vector of "hits" (hits), and a p-value adjustment method. It outputs a vector containing the size of the gene universe, the size of the gene set within this universe (i.e. how many genes from the universe map to this gene set), the total number of hits, the number of hits expected to occur in the gene set, the actual hits observed in the gene set, and the p-value from a hypergeometric test.

Usage

```
hyperGeoTest(geneSet, universe, hits)
```

Arguments

`geneSet`	a character vector specifying a gene set
`universe`	a character vector of all gene identifiers (usually all genes tested in a screen)
`hits`	a character vector of gene identifiers for those genes considered as hits

Value

a numeric vector containing the size of the gene universe (named by "Universe Size"), the size of the gene set within this universe (i.e. how many genes from the universe map to this gene set (named by "Gene Set Size"), the total number of hits (named by "Total Hits"), the number of hits expected to occur in the gene set (named by "Expected Hits"), the actual hits observed in the gene set (named by "Observed Hits"), and the pvalue from a hypergeometric test (named by "Pvalue").

Author(s)

John C. Rose, Xin Wang

See Also

multiHyperGeoTest

Examples

```
##example 1
gl <- runif(100, min=0, max=5)
gl <- order(gl)
names(gl) <- as.character(sample(x=seq(from=1, to=100, by=1), size=100,
replace=FALSE))
gs1 <- sample(names(gl), size=20, replace=FALSE)
gs2 <- sample(names(gl), size=20, replace=FALSE)
gsc <- list(subset1=gs1, subset2=gs2)
hypgeo <- hyperGeoTest(geneSet=gsc[["subset1"]], universe=names(gl),
hits=names(gl)[which(abs(gl) > 2)])
##example 2
library(org.Dm.eg.db)
library(KEGG.db)
##load phenotype vector (see the vignette for details about the
##preprocessing of this data set)
data("KcViab_Data4Enrich")
##Prepare kegg gene sets
DM_KEGG<-KeggGeneSets(species="Dm")
##Do the tests
hypgeoResults <- hyperGeoTest(geneSet=DM_KEGG[[1]], universe=
names(KcViab_Data4Enrich), hits=names(KcViab_Data4Enrich)[which(abs(
KcViab_Data4Enrich) > 2)])
```

---

interactome *Create an interactome from BioGRID data sets*

Description

This is a generic function.

When implemented as the S4 method of class NWA, this function creates an interactome before conducting network analysis.

To use this function for objects of class NWA:

interactome(object, interactionMatrix, species, link, reportDir = "HTSanalyzerReport", genetic=FALSE, verbose=TRUE)

Usage

```
interactome(object, ...)
```

Arguments

object	an object. When this function is implemented as the S4 method of class NWA, this argument is an object of class 'NWA'
...	other arguments (see below for the arguments supported by the method of class NWA)

interactionMatrix: an interaction matrix including columns 'InteractionType', 'InteractorA' and 'InteractorB'. If this matrix is available, the interactome can be directly built based on it.

species: a single character value specifying the species for which the data should be read. The current version supports one of the following species: "Dm" ("Drosophila_melanogaster"), "Hs" ("Homo_sapiens"), "Rn" ("Rattus_norvegicus"), "Mm" ("Mus_musculus"), "Ce" ("Caenorhabditis_elegans").

link: the link (url) where the data should be downloaded (in tab2 format). The default link is version 3.1.71 (valid on Dec. 5 2010).

reportDir: a single character value specifying the directory to store reports. The BioGRID data set will be downloaded and stored in a subdirectory called 'Data' in 'reportDir'.

genetic: a single logical value. If TRUE, genetic interactions will be kept; otherwise, they will be removed from the data set.

verbose: a single logical value indicating to display detailed messages (when verbose=TRUE) or not (when verbose=FALSE)

Details

This function provides two options to create an interactome for network analysis. The user can either input an interaction matrix including columns 'InteractionType', 'InteractionA' and 'InteractionB', or set 'species', 'link' and 'genetic' to download data set from BioGRID and extract corresponding interactions to build the interactome.

Another way to set up the interactome is to input a `graphNEL` object when the `NWA` object is created (i.e. nwa=new("NWA", pvalues, phenotypes, interactome)).

Value

In the end, this function will return an updated object with slot 'interactome' as an object of class `graphNEL`.

Author(s)

Xin Wang <xw264@cam.ac.uk>

See Also

`biogridDataDownload`, `NWA`

Examples

```
## Not run:
##example 1, input interactome when initializing an 'NWA' object
##load p-values and phenotypes
data("KcViab_PVals","KcViab_Data4Enrich")
##load BioGRID interactome for Drosophila Melanogaster
data("Biogrid_DM_Interactome")
##create a NWA (NetWork Analysis) object
nwa <- new("NWA",pvalues=KcViab_PVals, phenotypes=KcViab_Data4Enrich,
interactome=Biogrid_DM_Interactome)
##print nwa
nwa

library(BioNet)
```

```
##example 2, build an interactome from Biogrid data base
##create a NWA (NetWork Analysis) object
nwa <- new("NWA", pvalues=KcViab_PVals, phenotypes=KcViab_Data4Enrich)
##print nwa
nwa
##download data from BioGRID and build the interactome
nwa <- interactome(nwa, species="Dm", reportDir="NWATest")
##print nwa again
nwa

## End(Not run)
```

`KeggGeneSets` *Create a list of gene sets based on KEGG pathways terms*

Description

This function creates a list of gene sets based on KEGG pathways terms. It is species-specific, and returns a list of gene sets, each of which is a character vector of Entrez gene identifiers.

Usage

```
KeggGeneSets(species="Dm")
```

Arguments

`species`	a single character value specifying the species: "Dm" ("Drosophila_ melanogaster"), "Hs" ("Homo_sapiens"), "Rn" ("Rattus_norvegicus"), "Mm" ("Mus_musculus") or "Ce" ("Caenorhabditis_elegans"))

Details

This function relies on the following packages: GSEABase, KEGG.db.

Value

a list of gene sets, with names as KEGG pathway IDs. Each gene set is a character vector of Entrez gene identifiers.

Author(s)

Camille Terfve, Xin Wang

See Also

`GOGeneSets`

Examples

```
library(org.Dm.eg.db)
library(KEGG.db)
DM_KEGG<-KeggGeneSets(species = "Dm")
```

`mammalAnnotationConvertor` *Convert between different types of identifiers for mammalian species*

Description

This function converts an initial named data vector to the same vector but with a different identifier category (this function also works on matrices with gene identifiers as row names).

Usage

```
mammalAnnotationConvertor(geneList, initialIDs = "Entrez.gene", finalIDs =
"Entrez.gene", species = "Hs", keepMultipleMappings = TRUE, verbose=TRUE)
```

Arguments

`geneList`	a named integer or numeric vector, or a matrix with rows named by gene identifiers
`initialIDs`	a single character value specifying the type of initial identifiers for input 'geneList'. The current version can take one of the following types: "Ensembl.transcript", "Ensembl.prot", "Ensembl.gene", "Entrez.gene", "RefSeq", "Symbol" and "GenBank".
`finalIDs`	a single character value specifying the type of final identifiers to which users want to convert. The current version can take one of the following types: "Ensembl.transcript", "Ensembl.prot", "Ensembl.gene", "Entrez.gene", "RefSeq", "Symbol" and "GenBank"
`species`	a single character value specifying the species: "Hs" ("Homo_sapiens"), "Rn" ("Rattus_norvegicus") or "Mm" ("Mus_musculus")
`keepMultipleMappings`	a single logical value. If TRUE, the function keeps the entries with multiple mappings (first mapping is kept). If FALSE, the entries with multiple mappings will be discarded.
`verbose`	a single logical value indicating to display detailed messages (when verbose=TRUE) or not (when verbose=FALSE)

Details

This function removes the genes for which no mapping was found. This function relies on the org.Hs.eg.db/org.Mm.eg.db/org.Rn.eg.db packages and therefore only maps from any identifier to an Entrez gene ID or from an Entrez gene ID to any identifier.

Value

the same data vector/matrix but with another type of identifiers as names/row names

Author(s)

Camille Terfve, Xin Wang

See Also

drosoAnnotationConvertor, celAnnotationConvertor, annotationConvertor

Examples

```
##example 1: convert a named vector
library(org.Hs.eg.db)
x <- runif(10)
names(x) <- names(as.list(org.Hs.egSYMBOL2EG))[1:10]
xEntrez <- mammalAnnotationConvertor(geneList=x, initialIDs="Symbol",
finalIDs="Entrez.gene", species="Hs")
##example 2: convert a data matrix with row names as gene ids
library(org.Hs.eg.db)
x <- cbind(runif(10), runif(10))
rownames(x) <- names(as.list(org.Hs.egSYMBOL2EG))[1:10]
xEntrez <- mammalAnnotationConvertor(geneList=x, initialIDs="Symbol",
finalIDs="Entrez.gene", species="Hs")
```

---

multiHyperGeoTest *Hypergeometric tests on a list of gene sets*

Description

This function performs hypergeometric tests for over-representation of hits, on a list of gene sets. This function applies the function hyperGeoTest to an entire list of gene sets and returns a data frame.

Usage

```
multiHyperGeoTest(collectionOfGeneSets, universe, hits, minGeneSetSize =
15, pAdjustMethod = "BH", verbose = TRUE)
```

Arguments

collectionOfGeneSets	a list of gene sets, each of which is a character vector of gene identifiers
universe	a character vector of all gene identifiers (usually all genes tested in a screen)
hits	a character vector of gene identifiers for those considered as hits
minGeneSetSize	a single integer or numeric value specifying the minimum number of elements in a gene set that must map to elements of the gene universe. Gene sets with fewer genes than this number are removed from both hypergeometric analysis and GSEA.
pAdjustMethod	a single character value specifying the p-value adjustment method to be used (see 'p.adjust' for details)
verbose	a single logical value indicating to display detailed messages (when verbose=TRUE) or not (when verbose=FALSE)

Value

a data frame containing the results of the hypergeometric test (one row per gene set)

Author(s)

John C. Rose, Xin Wang

See Also

hyperGeoTest

Examples

```
##example 1
gl <- runif(100, min=0, max=5)
gl <- gl[order(gl, decreasing=TRUE)]
names(gl) <- as.character(sample(x=seq(from=1, to=100, by=1), size=100,
replace=FALSE))
gs1 <- sample(names(gl), size=20, replace=FALSE)
gs2 <- sample(names(gl), size=20, replace=FALSE)
gsc <- list(subset1=gs1, subset2=gs2)
hypgeo<-multiHyperGeoTest(collectionOfGeneSets=gsc, universe=names(gl),
hits=names(gl)[which(abs(gl) > 2)], minGeneSetSize = 2, pAdjustMethod ="BH")
##example 2
## Not run:
library(org.Dm.eg.db)
library(KEGG.db)
##load phenotype vector (see the vignette for details about the
##preprocessing of this data set)
data("KcViab_Data4Enrich")
DM_KEGG <- KeggGeneSets(species="Dm")
##Do multiple hypergeometric tests
hypgeoResults <- multiHyperGeoTest(collectionOfGeneSets=DM_KEGG,
universe=names(KcViab_Data4Enrich), hits=names(KcViab_Data4Enrich)[which(abs(
KcViab_Data4Enrich) > 2)], minGeneSetSize = 15, pAdjustMethod = "BH")

## End(Not run)
```

---

networkAnalysis *Identify enriched subnetworks*

Description

This function finds subnetworks enriched for genes with significant phenotypes based on the package 'BioNet'.

Usage

```
networkAnalysis(pvalues, graph, fdr=0.001, verbose=TRUE)
```

Arguments

pvalues	a numeric vector of p-values
graph	an object of class graphNEL, used as the interactome in the network analysis
fdr	a single numeric value specifying the false discovery for the scoring of nodes (see BioNet::scoreNodes and Dittrich et al., 2008 for details)
verbose	a single logical value indicating to display detailed messages (when verbose=TRUE) or not (when verbose=FALSE)

Details

This function takes in a vector of p-values and a graph standing for the interactome to identify the maximum scoring subnetwork based on the BioNet package.

Value

a subnetwork module of class `graphNEL`

Author(s)

Camille Terfve, Xin Wang

References

Beisser D, Klau GW, Dandekar T, Muller T, Dittrich MT. BioNet: an R-Package for the functional analysis of biological networks. Bioinformatics. 2010 Apr 15;26(8):1129-30.

Dittrich MT, Klau GW, Rosenwald A., Dandekar T and Muller T. *Identifying functional modules in protein-protein interaction networks: an integrated exact approach.* Bioinformatics 2008 24(13):i223-i231.

See Also

`networkPlot`, `viewSubNet`, `plotSubNet`

Examples

```
## Not run:
library(BioNet)
##load pvalues (see the vignette for details about the preprocessing of
##this data set)
data("KcViab_PVals")
##load interactome
data("Biogrid_DM_Interactome")
##identify subnetworks
enrichedSubNet <- networkAnalysis(pvalues=KcViab_PVals,
graph=Biogrid_DM_Interactome, fdr=0.001, verbose=TRUE)

## End(Not run)
```

`networkPlot` *Plot the enriched subnetwork*

Description

This function takes in a subnetwork module resulted from the function `networkAnalysis`, a vector of labels for nodes in the module and a phenotype vector (optional) to generate a figure.

Usage

```
networkPlot(nwAnalysisOutput, phenotypeVector=NULL)
```

Arguments

`nwAnalysisOutput`
: a list consisting of 'subnw' and 'labels', in which 'subnw' is the subnetwork module generated by the function `networkAnalysis`, while 'labels' is a character vector specifying the labels for all nodes in the module.

`phenotypeVector`
: a numeric or integer vector characterizing the phenotypes of nodes in the subnetwork module.

Details

The 'phenotypeVector' argument is optional. The subnetwork figure will be more readable if it is provided. See the function `plotModule` function in the 'BioNet' package for more details.

Value

a subnetwork module of class `graphNEL`

Author(s)

Xin Wang, Camille Terfve

References

Beisser D, Klau GW, Dandekar T, Muller T, Dittrich MT. BioNet: an R-Package for the functional analysis of biological networks. Bioinformatics. 2010 Apr 15;26(8):1129-30.

Dittrich MT, Klau GW, Rosenwald A., Dandekar T and Muller T. *Identifying functional modules in protein-protein interaction networks: an integrated exact approach.* Bioinformatics 2008 24(13):i223-i231.

See Also

`networkAnalysis`, `viewSubNet`, `plotSubNet`

Examples

```
## Not run:
library(BioNet)
library(org.Dm.eg.db)
##load pvalues, interactome, and phenotype vector (see the vignette for
##preprocessing details about this dataset)
data("KcViab_PVals", "Biogrid_DM_Interactome", "KcViab_Data4Enrich")
##Identify subnetworks
enrichedSubNet <- networkAnalysis(pvalues=KcViab_PVals,
graph=Biogrid_DM_Interactome, fdr=0.001, verbose=TRUE)
dev.off()
map <- as.list(get("org.Dm.egSYMBOL"))
labels <- map[nodes(enrichedSubNet)]
nwAnalysisResult <- list(subnw=enrichedSubNet, labels=labels)
networkPlot(nwAnalysisOutput=nwAnalysisResult, phenotypeVector=
KcViab_Data4Enrich)

## End(Not run)
```

NWA-class *An S4 class for NetWork Analysis on high-throughput screens*

Description

This class includes a series of methods to do network analysis for high-throughput screens.

Objects from the Class

Objects of class NWA can be created from `new("NWA", pvalues, phenotypes=NULL, interactome=NULL)` (see the examples below)

Slots

`pvalues`: a numeric vector of p-values.

`phenotypes`: a numeric or integer vector of phenotypes.

`interactome`: an object of class `graphNEL`.

`fdr`: one parameter for BioNet to score nodes in the interactome.

`result`: a list consisting of subnetwork module identified by BioNet and a vector of labels for nodes of the subnetwork module.

`summary`: a list of summary information for `p-values`, `phenotypes`, `interactome` and `result`.

`preprocessed`: a logical value specifying whether or not input data has been preprocessed.

Methods

An overview of methods with class-specific functionality: More detailed introduction can be found in help for each specific function.

`preprocess` do preprocessing for the input vector of p-values and the vector of phenotypes including: a) removing NAs in `p-values` and `phenotypes`; b) invoking function `duplicateRemover` to process duplicated phenotypes and p-values (see `duplicateRemover` for more details); c) invoking function `annotationConvertor` to convert annotations;

`analyze` invoke function `networkAnalysis` to identify enriched sub- networks based on input parameter list `para`.

`summarize` print summary information about `p-values`, `phenotypes`, `interactome` and `result`.

`interactome` build an interactome for the network analysis.

`viewSubNet` plot a figure of identified subnetwork.

`plotSubNet` plot and save a figure of identified subnetwork.

`report` generate html reports.

Author(s)

Xin Wang <xw264@cam.ac.uk>

See Also

`preprocess` `analyze` `summarize` `interactome` `viewSubNet` `plotSubNet` `report`

Examples

```
## Not run:
library(BioNet)
##load p-values and phenotypes
data("KcViab_PVals","KcViab_Data4Enrich")
##load Biogrid interactome for Drosophila Melanogaster
data("Biogrid_DM_Interactome")
##create a NWA (NetWork Analysis) object
nwa <- new("NWA", pvalues=KcViab_PVals, phenotypes=KcViab_Data4Enrich,
interactome=Biogrid_DM_Interactome)
##preprocessing
nwa <- preprocess(nwa, species="Dm", initialIDs="Entrez.gene",
keepMultipleMappings=TRUE, duplicateRemoverMethod="max")
##To create an interactome
nwa <- interactome(nwa, species="Dm", reportDir="HTSanalyzerReport",
genetic=FALSE)
##do network analysis
nwa <- analyze(nwa, fdr=0.001, species="Dm")
graphics.off()
##view identified subnetwork
viewSubNet(nwa)
##report to html pages
report(object=nwa, experimentName="NWATest", species="Dm", allSig=TRUE,
keggGSCs="PW_KEGG", goGSCs=c("GO_BP", "GO_MF", "GO_CC"), reportDir=
"NWATestReport")
##browse the index page of the report
browseURL(file.path(getwd(), "NWATestReport", "index.html"))

## End(Not run)
```

---

pairwiseGseaPlot *Produce a plot for pairwise GSEA result on one gene set*

Description

This function plots results of the GSEA analyses on one gene set for two phenotypes in parallel .

Usage

```
pairwiseGseaPlot(gl1, gl2, geneSet, exponent=1, output="png", filepath,
filename, ...)
```

Arguments

gl1	a named numeric or integer vector where names are gene identifiers of the same type as the ones in the gene set collection, and values are the measurements on phenotype one corresponding to those genes. This vector MUST be ordered
gl2	a named numeric or integer vector where names are gene identifiers of the same type as the ones in the gene set collection, and values are the measurements on phenotype two corresponding to those genes. This vector MUST be ordered
geneSet	a character vector specifying a gene set (no names, just a vector of characters corresponding to the IDs)

exponent	a single numeric or integer value (set as 1 by default) specifying the exponent of the GSEA method.
output	a single character value specifying the format of output figure: "pdf" or "png"
filepath	a single character value specifying the directory where these plots will be stored
filename	a single character value specifying the name of the gene set for which the plot is produced
...	other arguments of the function pdf or png

Details

The plots are going to be produced and stored in the directory 'filepath' with the name 'filename'.

Author(s)

Camille Terfve, Xin Wang

References

Subramanian, A., Tamayo, P., Mootha, V. K., Mukherjee, S., Ebert, B. L., Gillette, M. A., Paulovich, A., Pomeroy, S. L., Golub, T. R., Lander, E. S. & Mesirov, J. P. (2005) *Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles.* Proc. Natl. Acad. Sci. USA 102, 15545-15550.

See Also

pairwiseGsea, pairwisePhenoMannWhit

Examples

```
## Not run:
gl1 <- runif(100, min=-5, max=5)
gl1 <- gl1[order(gl1, decreasing=TRUE)]
gl2 <- runif(100, min=-5, max=5)
gl2 <- gl2[order(gl2, decreasing=TRUE)]
names(gl1) <- as.character(sample(x=seq(from=1, to=100, by=1), size=100,
replace=FALSE))
names(gl2) <- names(gl1)
gs1 <- sample(names(gl1), size=20, replace=FALSE)
gs2 <- sample(names(gl1), size=20, replace=FALSE)
gsc <- list(subset1=gs1, subset2=gs2)
pairwiseGseaPlot(gl1=gl1, gl2=gl2, geneSet=gsc[["subset1"]], filepath=".",
filename="geneSet.pdf", output="pdf", width=8, height=6)

## End(Not run)
```

`pairwiseGsea` *GSEA on a pair of phenotypes*

Description

This function performs pairwise GSEA: it looks for gene sets that are specifically over-represented towards the two different ends of two ranked phenotype vectors, in a gene set collection.

Usage

```
pairwiseGsea(gl1, gl2, gsc, exponent=1, nPermutations=1000,
minGeneSetSize=15, pAdjustMethod="BH")
```

Arguments

`gl1`	a named numeric or integer vector where names are gene identifiers of the same type as the ones in the gene set collection, and values are the measurements on phenotype one corresponding to those genes. This vector MUST be ordered (decreasing or increasing)
`gl2`	a named numeric or integer vector where names are gene identifiers of the same type as the ones in the gene set collection, and values are the measurements on phenotype two corresponding to those genes. This vector MUST be ordered
`gsc`	a list of gene sets, each of which in the list is a character vector of gene identifiers.
`exponent`	a single numeric or integer value (set as 1 by default) specifying the exponent of the GSEA method.
`nPermutations`	a single numeric or integer value specifying the number of permutation tests for each gene set
`minGeneSetSize`	a single numeric or integer value specifying the minimum size required for a gene set to be considered.
`pAdjustMethod`	a single character value specifying the p-value adjustment method to be used (see 'p.adjust' for details)

Details

phenotypes one and two must be measured on the same genes, i.e. the two vectors gl1 and gl2 must have the same length and their names must match, but the two vectors must be ordered separately, i.e.one phenotype vector is ordered based on the values of that phenotype only

Value

a table with a row for each gene set, containing the p-values for the GSEA, and the observed scores for each of the phenotypes independently. The table is ordered by the p-value column.

Author(s)

Camille Terfve, Xin Wang

References

Subramanian, A., Tamayo, P., Mootha, V. K., Mukherjee, S., Ebert, B. L., Gillette, M. A., Paulovich, A., Pomeroy, S. L., Golub, T. R., Lander, E. S. & Mesirov, J. P. (2005) *Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles.* Proc. Natl. Acad. Sci. USA 102, 15545-15550.

See Also

pairwiseGseaPlot, pairwisePhenoMannWhit

Examples

```
gl1 <- runif(100, min=-5, max=5)
gl1 <- gl1[order(gl1, decreasing=TRUE)]
gl2 <- runif(100, min=-5, max=5)
gl2 <- gl2[order(gl2, decreasing=TRUE)]
names(gl1) <- as.character(sample(x=seq(from=1, to=100, by=1), size=100,
replace=FALSE))
names(gl2) <- names(gl1)
gs1 <- sample(names(gl1), size=20, replace=FALSE)
gs2 <- sample(names(gl1), size=20, replace=FALSE)
gsc <- list(subset1=gs1, subset2=gs2)
pwGSEAscore <- pairwiseGsea(gl1=gl1, gl2=gl2, gsc=gsc)
```

---

pairwisePhenoMannWhit

Mann-Whitney U test for shift in location of genes from gene sets on a pair of phenotypes

---

Description

This function performs a Mann-Whitney U test for shift in location of genes from gene sets, on a pair of phenotypes. It looks for gene sets that are represented towards the two different ends of two ranked lists of genes, i.e. whose phenotype distribution is located around two different ends of the two phenotype vectors, rather than spread on the whole list in both lists.

Usage

```
pairwisePhenoMannWhit(gl1, gl2, gsc, minGeneSetSize=15, pAdjustMethod="BH")
```

Arguments

gl1	a named numeric or integer vector where names are gene identifiers of the same type as the ones in the gene set collection, and values are the measurements on phenotype one corresponding to those genes. This vector MUST be ordered (decreasing or increasing)
gl2	a named numeric or integer vector where names are gene identifiers of the same type as the ones in the gene set collection, and values are the measurements on phenotype two corresponding to those genes. This vector MUST be ordered
gsc	a list of gene sets, each of which in the list is a character vector of gene identifiers.

`minGeneSetSize`
: a single numeric or integer value specifying the minimum size required for a gene set to be considered.

`pAdjustMethod`
: a single character value specifying the p-value adjustment method to be used (see 'p.adjust' for details)

Value

a table with a row for each gene set, containing the p-value for the Mann-Whitney U test, and the adjusted p-value. The table is ordered by the p-value column.

Author(s)

Camille Terfve, Xin Wang

See Also

`pairwiseGseaPlot`, `pairwiseGsea`

Examples

```
gl1 <- runif(100, min=-5, max=5)
gl2 <- runif(100, min=-5, max=5)
names(gl1) <- as.character(sample(x=seq(from=1, to=100, by=1), size=100,
replace=FALSE))
names(gl2) <- names(gl1)
gs1 <- sample(names(gl1), size=20, replace=FALSE)
gs2 <- sample(names(gl1), size=20, replace=FALSE)
gsc <- list(subset1=gs1, subset2=gs2)
pwGSC <- pairwisePhenoMannWhit(gl1=gl1, gl2=gl2, gsc=gsc, minGeneSetSize=2)
```

---

permutationPvalueCollectionGsea

Compute the GSEA p-values for a list of gene sets

Description

Compute the nominal p-value associated with a GSEA for a list of gene sets, from the outputs of the function `collectionGsea`.

Usage

```
permutationPvalueCollectionGsea(permScores, dataScores)
```

Arguments

`permScores`
: a numeric matrix of permutation-based scores, of which each row is named and corresponds to a gene set, output from the function `collectionGsea`

`dataScores`
: a named numeric vector of observed scores output from the function `collectionGsea`

Value

a named numeric vector of p-values, each of which corresponds to a gene set

Author(s)

Camille Terfve, Xin Wang

References

Subramanian, A., Tamayo, P., Mootha, V. K., Mukherjee, S., Ebert, B. L., Gillette, M. A., Paulovich, A., Pomeroy, S. L., Golub, T. R., Lander, E. S. & Mesirov, J. P. (2005) *Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles.* Proc. Natl. Acad. Sci. USA 102, 15545-15550.

See Also

collectionGsea

Examples

```
##example 1
gl <- runif(100, min=0, max=5)
gl <- gl[order(gl, decreasing=TRUE)]
names(gl) <- as.character(sample(x=seq(from=1, to=100, by=1), size=100,
replace=FALSE))
gs1 <- sample(names(gl), size=20, replace=FALSE)
gs2 <- sample(names(gl), size=20, replace=FALSE)
gsc <- list(subset1=gs1, subset2=gs2)
GSCscores <- collectionGsea(collectionOfGeneSets=gsc, geneList=gl,
exponent=1, nPermutations=1000, minGeneSetSize=5)
GSCpvals <- permutationPvalueCollectionGsea(permScores=
GSCscores$Permutation.scores, dataScores=GSCscores$Observed.scores)
##example 2
## Not run:
library(KEGG.db)
library(org.Dm.eg.db)
##load phenotype vector (see the vignette for details about the
##preprocessing of this data set)
data("KcViab_Data4Enrich")
DM_KEGG <- KeggGeneSets(species="Dm")
GSCscores <- collectionGsea(collectionOfGeneSets=DM_KEGG, geneList=
KcViab_Data4Enrich, exponent=1, nPermutations=1000, minGeneSetSize=100)
GSCpvals <- permutationPvalueCollectionGsea(permScores=
GSCscores$Permutation.scores, dataScores=GSCscores$Observed.scores)

## End(Not run)
```

`plotEnrichMap` *Plot and save an enrichment map for results of GSEA or hypergeometric tests*

Description

This is a generic function.

When implemented as the S4 method for objects of class GSCA, this function will plot and save an enrichment map for GSEA or Hypergeometric test results.

To use this function for objects of class GSCA:

plotEnrichMap(object, resultName="GSEA.results", gscs, ntop=NULL, allSig=TRUE, gsNameType="id", displayEdgeLabel=TRUE, layout= "layout.fruchterman.reingold", filepath=".", filename="test.png", output="png", ...)

Usage

```
plotEnrichMap(object, ...)
```

Arguments

object	an object. When this function is implemented as the S4 method of class GSCA, this argument is an object of class GSCA.
...	other arguments. (see below for the arguments supported by the method of class GSCA)

resultName: a single character value: 'HyperGeo.results' or 'GSEA.results'

gscs: a character vector specifying the names of gene set collections of which the top significant gene sets will be plotted

ntop: a single integer or numeric value indicating how many gene sets of top significance will be plotted.

allSig: a single logical value. If 'TRUE', all significant gene sets (adjusted p-value < 'pValueCutoff' of slot 'para') will be used; otherwise, only top 'ntop' gene sets will be used.

gsNameType: a single character value specifying the type of the gene set names that will be displayed as the names of nodes in the enrichment map. The type of the gene set names should be one of the following: "id", "term" or "none".

displayEdgeLabel: a single logical value specifying whether or not to display the labels of the edges in the enrichment map

layout: a single character value specifying the layout of the enrichment map. (see help(layout) of the package igraph)

filepath: a single character value specifying where to store the enrichment map.

output: a single character value specifying the format of output image: "pdf" or "png"

... other arguments used by the function png or pdf such as 'width' and 'height'

Details

See help(viewEnrichMap) for more details about the enrichment map for GSEA.

Author(s)

Xin Wang <xw264@cam.ac.uk>

See Also

viewEnrichMap

Examples

```
## Not run:
library(org.Dm.eg.db)
library(KEGG.db)
##load data for enrichment analyses
data("KcViab_GSCA")
##plot and save the enrichment map
plotEnrichMap(KcViab_GSCA, gscs=c("GO_MF"), allSig=TRUE, ntop=NULL, gsNameType="id",
displayEdgeLabel=FALSE,layout="layout.fruchterman.reingold", filepath="~",
filename="GO_MF.pdf",output="pdf", width=8, height=8)

## End(Not run)
```

`plotGSEA` *Plot and save figures of GSEA results for top significant gene sets*

Description

This is a generic function.

When implemented as the S4 method for objects of class GSCA, this function plots figures of the positions of genes of the gene set in the ranked gene list and the location of the enrichment score for top significant gene sets.

To use this function for objects of class GSCA:

plotGSEA(object, gscs, ntop=NULL, allSig=FALSE, filepath=".", output= "png", ...)

Usage

```
plotGSEA(object, ...)
```

Arguments

`object`	an object. When this function is implemented as the S4 method of class GSCA, this argument is an object of class GSCA.
`...`	other arguments. (see below for the arguments supported by the method of class GSCA)

gscs: a character vector specifying the names of gene set collections whose top significant gene sets will be plotted

ntop: a single integer or numeric value specifying how many gene sets of top significance will be plotted.

allSig: a single logical value. If 'TRUE', all significant gene sets (GSEA adjusted p-value < 'pValueCutoff' of slot 'para') will be plotted; otherwise, only top 'ntop' gene sets will be plotted.

filepath: a single character value specifying where to store GSEA figures.

output: a single character value specifying the format of output image: "pdf" or "png"

... other arguments used by the function `png` or `pdf` such as 'width' and 'height'

Details

To make GSEA plots of top significance using this function, the user can only choose one method: either assign an integer to the argument 'ntop' or set the argument 'allSig' to 'TRUE'. Exceptions will occur if both methods are used, or no method is used. Please also note that the argument 'ntop' is a cutoff for all gene set collections in the argument 'gscs'.

We suggest to perform `summarize(gsca, what="Result")` first to have an idea of how many significant gene sets there are, and then choose to plot them all or just the top ones.

Author(s)

Xin Wang <xw264@cam.ac.uk>

See Also

viewGSEA, gseaPlots

Examples

```
## Not run:
library(org.Dm.eg.db)
library(KEGG.db)
##load data for enrichment analyses
data("KcViab_Data4Enrich")
##select hits
hits <- names(KcViab_Data4Enrich)[which(abs(KcViab_Data4Enrich) > 2)]
##set up a list of gene set collections
PW_KEGG <- KeggGeneSets(species = "Dm")
gscList <- list(PW_KEGG = PW_KEGG)
##create an object of class 'GSCA'
gsca <- new("GSCA", listOfGeneSetCollections=gscList, geneList =
KcViab_Data4Enrich, hits = hits)
##print summary of gsca
summarize(gsca)
##do preprocessing (KcViab_Data4Enrich has already been preprocessed)
gsca <- preprocess(gsca, species="Dm", initialIDs = "Entrez.gene",
keepMultipleMappings = TRUE, duplicateRemoverMethod = "max",
orderAbsValue = FALSE)
##print summary of gsca again
summarize(gsca)
##do hypergeometric tests and GSEA
gsca <- analyze(gsca, para = list(pValueCutoff = 0.05, pAdjustMethod
= "BH", nPermutations = 1000, minGeneSetSize = 100, exponent = 1))
##print summary of results
summarize(gsca, what="Result")
##plot all significant gene sets
plotGSEA(gsca, gscs=c("PW_KEGG"), allSig=TRUE, filepath=".", output=
"pdf", width=8, height=8)

## End(Not run)
```

plotSubNet *Plot and save a figure of the enriched subnetwork*

Description

This is an generic function.

When implemented as the S4 method for class NWA, this function invokes the function networkPlot to plot and save the subnetwork identified by the 'BioNet' package.

To use this function for objects of class NWA:

plotSubNet(object, filepath, filename, output, ...)

Usage

```
plotSubNet(object, ...)
```

Arguments

object	an object. When implemented as S4 methods of class NWA, this argument is an object of class NWA.
...	other arguments. (see below for the arguments supported by the method of class NWA)

filepath: a single character value specifying the directory where the figure will be stored

filename: a single character value specifying the name of the figure to save

output: a single character value specifying the format of output image: "pdf" or "png"

...: all other arguments for the function pdf or png such as the argument 'width' and 'height'

Details

After the analyses step for an object of class 'NWA', users can generate the enriched subnetwork identified by the 'BioNet' package. If the slot 'phenotype' was inputted when initializing the object, this function will send it to the function networkPlot as the argument phenotypeVector to highlight nodes in different colors. If the argument species of the function analyze has been assigned, labels of nodes of this subnetwork will be mapped to gene symbols corresponding to the species; otherwise, Entrez identifiers will be used as the labels.

Author(s)

Xin Wang <xw264@cam.ac.uk>

See Also

networkPlot, viewSubNet, analyze

Examples

```
## Not run:
library(BioNet)
##load p-values and phenotypes
data("KcViab_Data4Enrich","KcViab_PVals")
##load Biogrid interactome for Drosophila Melanogaster
data("Biogrid_DM_Interactome")
##create a NWA (NetWork Analysis) object
nwa <- new("NWA", pvalues=KcViab_PVals, phenotypes=KcViab_Data4Enrich,
interactome=Biogrid_DM_Interactome)
##preprocessing
nwa <- preprocess(nwa, species="Dm", initialIDs="Entrez.gene",
keepMultipleMappings=TRUE, duplicateRemoverMethod="max")
##To create an interactome:
##nwa<-interactome(nwa, species="Dm", reportDir="HTSanalyzerReport",
##genetic=FALSE)
##do network analysis
nwa <- analyze(nwa, fdr=0.001, species="Dm")
graphics.off()
##plot and save the identified subnetwork
plotSubNet(nwa, filepath=".", filename="subnetwork.pdf",
output="pdf", width=8, height=8)

## End(Not run)
```

---

preprocess *A preprocessing method for objects of class GSCA or NWA*

Description

This is a generic function.

When implemented as the S4 method for objects of class GSCA or NWA, this function filters out invalid data, removes duplicated genes, converts annotations to Entrez identifiers, etc.

To use this function for objects of class GSCA:

preprocess(object, species="Dm", initialIDs="FlybaseCG", keepMultipleMappings =TRUE, duplicateRemoverMethod="max", orderAbsValue=FALSE, verbose=TRUE)

To use this function for objects of NWA:

preprocess(object, species="Dm", initialIDs="FlybaseCG", keepMultipleMappings =TRUE, duplicateRemoverMethod="max", verbose=TRUE)

Usage

```
preprocess(object, ...)
```

Arguments

object	an object. When this function is implemented as the S4 method of class GSCA or NWA, this argument is an object of class 'GSCA' or NWA.
...	other arguments depending on class (see below for the arguments supported by class GSCA and/or NWA)

species: a single character value specifying the species for which the data should be read. The current version supports one of the following species: "Dm" ("Drosophila_melanogaster"), "Hs" ("Homo_sapiens"), "Rn" ("Rattus_ norvegicus"), "Mm" ("Mus_musculus"), "Ce" ("Caenorhabditis_elegans"). This is an optional argument here. If it is provided, then the labels of nodes of the identified subnetwork will be mapped from Entrez IDs to gene symbols; otherwise, Entrez IDs will be used as labels for those nodes.

initialIDs: a single character value specifying the type of initial identifiers for input 'geneList'. Current version can take one of the following types: "Ensembl.transcript", "Ensembl.prot", "Ensembl.gene", "Entrez.gene", "RefSeq", "Symbol" and "GenBank" for all supported species; "Flybase", "FlybaseCG" and "FlybaseProt" in addition for Drosophila Melanogaster; "wormbase" in addition for Caenorhabditis Elegans.

keepMultipleMappings: a single logical value. If 'TRUE', the function keeps the entries with multiple mappings (first mapping is kept). If 'FALSE', the entries with multiple mappings will be discarded.

duplicateRemoverMethod: a single character value specifying the method to remove the duplicates (should the minimum, maximum or average observation for a same construct be kept). Current version provides "min" (minimum), "max" (maximum), "average" and "fc.avg" (fold change average). The minimum and maximum should be understood in terms of absolute values (i.e. min/max effect, no matter the sign). The fold change average method converts the fold changes to ratios, averages them and converts the average back to a fold change.

orderAbsValue: a single logical value indicating whether the values should be converted to absolute values and then ordered (if TRUE), or ordered as they are (if FALSE). This argument is only for class `GSCA`.

verbose: a single logical value suggesting to display detailed messages (when verbose=TRUE) or not (when verbose=FALSE)

Details

This function will do the following preprocessing steps:

1: filter out p-values (the slot `pvalues` of class `NWA`), phenotypes (the slot `phenotypes` of class `NWA`) and data for enrichment (the slot `geneList` of class `GSCA`) with NA values or without valid names, and invalid gene names (the slot `hits` of class `GSCA`);

2: invoke function `duplicateRemover` to remove duplicated genes in the slot `pvalues`, `phenotypes` of class `NWA`, and the slot `geneList` and `hits` of class `GSCA`;

3: invoke function `annotationConvertor` to convert annotations from `initialIDs` to Entrez identifiers. Please note that the slot `hits` and the names of the slot `geneList` of class `GSCA`, the names of the slot `pvalues` and the names of the slot `phenotypes` of class `NWA` must have the same type of gene annotation specified by `initialIDs`;

4: order the data for enrichment decreasingly for objects of class `GSCA`.

See the function `duplicateRemover` for more details about how to remove duplicated genes.

See the function `annotationConvertor` for more details about how to convert annotations.

Value

In the end, this function will return an updated object of class `GSCA` or `NWA`.

Author(s)

Xin Wang <xw264@cam.ac.uk>

See Also

duplicateRemover, annotationConvertor

Examples

```
## Not run:
library(org.Dm.eg.db)
library(KEGG.db)
##load data for enrichment analyses
data("KcViab_Data4Enrich")
##select hits
hits <- names(KcViab_Data4Enrich)[which(abs(KcViab_Data4Enrich) > 2)]
##set up a list of gene set collections
PW_KEGG <- KeggGeneSets(species = "Dm")
gscList <- list(PW_KEGG = PW_KEGG)
##create an object of class 'GSCA'
gsca <- new("GSCA", listOfGeneSetCollections=gscList, geneList =
KcViab_Data4Enrich, hits = hits)
##print gsca
summarize(gsca, what = c("GeneList", "Hits"))
##do preprocessing (KcViab_Data4Enrich has already been preprocessed)
gsca <- preprocess(gsca, species="Dm", initialIDs = "Entrez.gene",
keepMultipleMappings = TRUE, duplicateRemoverMethod = "max",
orderAbsValue = FALSE)
##print updated object
summarize(gsca, what = c("GeneList", "Hits"))

## End(Not run)
```

---

reportAll *Write HTML reports for both the enrichment and network analyses*

Description

This is a generic function.

When implemented as the method of class GSCA and NWA, this function produces a report for both the results of Gene Set Collection Analysis and the Network Analysis.

To use reportAll for objects of class GSCA and NWA:

reportAll(gsca, nwa, experimentName="Unknown", species=NULL, ntop=NULL, allSig=FALSE, keggGSCs=NULL, goGSCs=NULL, reportDir="HTSanalyzerReport")

Usage

```
reportAll(gsca, nwa, ...)
```

Arguments

gsca	an object of class GSCA (see help(GSCA))
nwa	an object of class NWA (see help(NWA))
...	other arguments. (see below for the arguments supported by the method of class NWA and GSCA)

experimentName: a single character value specifying the name of the experiment (just for you own record)

species: a single character value specifying the species for which the data should be read. The current version supports one of the following species: "Dm" ("Drosophila_melanogaster"), "Hs" ("Homo_sapiens"), "Rn" ("Rattus_ norvegicus"), "Mm" ("Mus_musculus"), "Ce" ("Caenorhabditis_elegans").

ntop: a single integer value specifying the number of plots to be produced for the GSEA analysis. For each gene set collection, plots are produced for the top 'ntop' most significant p-values.

allSig: a single logical value determining whether or not to generate plots for all significant gene sets. A gene set is significant if its corresponding adjusted p-value is less than the pValueCutoff set in function analyze. (see function analyze for more details)

keggGSCs: a character vector of names of all KEGG gene set collections. This will help create web links for KEGG terms.

goGSCs: a character vector of names of all GO gene set collections. This will help create web links for GO terms.

reportDir: a single character value specifying the directory to store reports

Details

This function takes in the objects of the two wrapper classes (GSCA and NWA) and writes a report into the user-specified directory. An index HTML file containing a summary of all results and hyperlinked tabs to more detailed results will be generated in the root directory. The other HTML files will be stored in a subdirectory called 'html'. All images including GSEA plots and subnetwork figure will be produced in a sub- directory called 'image'. All documents or text files such as the files containing significant gene sets of the hypergeometric test results will be stored in a subdirectory called 'doc'.

Author(s)

Xin Wang, Camille Terfve

See Also

report, writeReportHTSA

Examples

```
## Not run:
##(see the vignette for details about the preprocessing of this data set)
library(KEGG.db)
library(GO.db)
library(AnnotationDbi)
data("KcViab_GSCA")
data("KcViab_NWA")
##append gene set terms to results
```

```
KcViab_GSCA<-appendGSTerms(KcViab_GSCA, keggGSCs="PW_KEGG",
goGSCs=c("GO_BP","GO_MF","GO_CC"))
##report both analyses
reportAll(gsca=KcViab_GSCA, nwa=KcViab_NWA, experimentName="KcViab",
species="Dm", allSig=TRUE, keggGSCs="PW_KEGG", goGSCs=c("GO_BP","GO_MF",
"GO_CC"), reportDir="HTSanalyzerReport")
browseURL(file.path(getwd(), "HTSanalyzerReport", "index.html"))

## End(Not run)
```

`report` *Write HTML reports for enrichment or network analyses*

Description

This is a generic function.

When implemented as the method of class `GSCA` or `NWA`, this function produces reports for either the Gene Set Collection Analysis or the Network Analysis.

To use `report` for objects of class `GSCA` or `NWA`:

report(object, experimentName="Unknown", species=NULL, ntop=NULL, allSig=FALSE, keggGSCs=NULL, goGSCs=NULL, reportDir="HTSanalyzerReport")

Usage

```
report(object, ...)
```

Arguments

`object`	an object. When implemented as S4 methods of class `GSCA` or `NWA`, this argument is either an object of class `GSCA` or `NWA`.
`...`	other arguments. (see below for the arguments supported by the method of class `GSCA` or `NWA`)

experimentName: a single character value specifying the name of the experiment (just for you own record)

species: a single character value specifying the species for which the data should be read. The current version supports one of the following species: "Dm" ("Drosophila_melanogaster"), "Hs" ("Homo_sapiens"), "Rn" ("Rattus_ norvegicus"), "Mm" ("Mus_musculus"), "Ce" ("Caenorhabditis_elegans").

ntop: a single integer value specifying the number of plots to be produced for the GSEA results. For each gene set collection, plots are produced for the 'ntop' most significant gene sets.

allSig: a single logical value indicating whether or not to generate plots for all significant gene sets. A gene set is significant if its corresponding adjusted p-value is less than the `pValueCutoff` set in the function `analyze`. (see `help(analyze)` for more details)

keggGSCs: a character vector of names of all KEGG gene set collections. This will help create web links for KEGG terms.

goGSCs: a character vector of names of all GO gene set collections. This will help create web links for GO terms.

reportDir: a single character value specifying the directory to store reports

Details

This function take in the objects of the two wrapper classes (GSCA and NWA) and writes a report into the user-specified directory. An index HTML file containing a summary of all results and hyperlinked tabs to more detailed results will be generated in the root directory. The other HTML files will be stored in a subdirectory called 'html'. All images including GSEA plots, enrichment maps and the subnetwork figure will be produced in a subdirectory called 'image'. All documents or text files such as the files containing significant gene sets of the hyper- geometric test results will be stored in a subdirectory called 'doc'.

Author(s)

Xin Wang, Camille Terfve

See Also

reportAll, writeReportHTSA

Examples

```
## Not run:
##(see the vignette for details about the preprocessing of this data set)
library(KEGG.db)
library(GO.db)
library(AnnotationDbi)
##report for a GSCA object
data("KcViab_GSCA")
##append gene set terms to results
KcViab_GSCA<-appendGSTerms(KcViab_GSCA, keggGSCs="PW_KEGG",
goGSCs=c("GO_BP","GO_MF","GO_CC"))
report(object=KcViab_GSCA, experimentName="KcViab", species="Dm",
allSig=TRUE, keggGSCs="PW_KEGG", goGSCs=c("GO_BP","GO_MF","GO_CC"),
reportDir="HTSanalyzerGSCAReport")
browseURL(file.path(getwd(), "HTSanalyzerGSCAReport", "index.html"))
##report for a NWA object
data("KcViab_NWA")
report(object=KcViab_NWA, experimentName="KcViab", species="Dm",
allSig=TRUE, keggGSCs="PW_KEGG", goGSCs=c("GO_BP","GO_MF","GO_CC"),
reportDir="HTSanalyzerNWReport")
browseURL(file.path(getwd(), "HTSanalyzerNWReport", "index.html"))

## End(Not run)
```

---

summarize *Print summary information for an object of class GSCA or NWA*

Description

This is a generic function.

When implemented as the S4 method for objects of class GSCA or NWA, this function prints a summary of information about the slots of these classes.

To use this function for objects of class GSCA or NWA:

summarize(object, what="ALL")

Usage

```
summarize(object, ...)
```

Arguments

object	an object. When this function is implemented as the S4 method of class GSCA or NWA, this argument is an object of class GSCA or NWA.
...	other arguments depending on class (see below for the arguments supported by the method of class GSCA and NWA)
	what: a single character value or a character vector of key words specifying what to print (see details below).

Details

For an object of class GSCA, the key words are 'GSC' (the slot 'listOfGeneSetCollections'), 'GeneList' (the slot 'geneList'), 'Hits' (the slot 'hits'), 'Para' (the slot 'para'), 'Result' (the slot 'result') and 'ALL' (all slots).

For an object of class NWA, the key words include 'Pval' (the slot 'pvalue'), 'Phenotype' (the slot 'phenotype'), 'Interactome' (the slot 'interactome'), 'Para' (the slot 'fdr'), 'Result' (the slot 'result') and 'ALL' (all slots).

Author(s)

Xin Wang <xw264@cam.ac.uk>

Examples

```
## Not run:
library(org.Dm.eg.db)
library(KEGG.db)
##load data for enrichment analyses
data("KcViab_Data4Enrich")
##select hits
hits <- names(KcViab_Data4Enrich)[which(abs(KcViab_Data4Enrich) > 2)]
##set up a list of gene set collections
PW_KEGG <- KeggGeneSets(species = "Dm")
gscList <- list(PW_KEGG = PW_KEGG)
##create an object of class 'GSCA'
gsca <- new("GSCA", listOfGeneSetCollections=gscList, geneList =
KcViab_Data4Enrich, hits = hits)
##print summary of gsca
summarize(gsca)
##do preprocessing (KcViab_Data4Enrich has already been preprocessed)
gsca <- preprocess(gsca, species="Dm", initialIDs = "Entrez.gene",
keepMultipleMappings = TRUE, duplicateRemoverMethod = "max",
orderAbsValue = FALSE)
##print summary of gsca again
summarize(gsca)
##do hypergeometric tests and GSEA
gsca <- analyze(gsca, para = list(pValueCutoff = 0.05, pAdjustMethod
= "BH", nPermutations = 1000, minGeneSetSize = 100, exponent = 1))
##print summary of results
summarize(gsca, what="Result")

## End(Not run)
```

`viewEnrichMap` *Plot a figure of the enrichment map for GSEA or Hypergeometric tests*

Description

This is a generic function.

When implemented as the S4 method for objects of class GSCA, this function will plot an enrichment map for GSEA or Hypergeometric test results.

To use this function for objects of class GSCA:

viewEnrichMap(object, resultName="GSEA.results", gscs, ntop=NULL, allSig=TRUE, gsNameType="id", displayEdgeLabel=TRUE, layout= "layout.fruchterman.reingold")

Usage

```
viewEnrichMap(object, ...)
```

Arguments

`object`	an object. When this function is implemented as the S4 method of class GSCA, this argument is an object of class GSCA.
`...`	other arguments. (see below for the arguments supported by the method of class GSCA)

resultName: a single character value: 'HyperGeo.results' or 'GSEA.results'

gscs: a character vector specifying the names of gene set collections of which the top significant gene sets will be plotted

ntop: a single integer or numeric value specifying how many gene sets of top significance will be plotted.

allSig: a single logical value. If 'TRUE', all significant gene sets (GSEA adjusted p-value < 'pValueCutoff' of slot 'para') will be used; otherwise, only top 'ntop' gene sets will be used.

gsNameType: a single character value specifying the type of the gene set names that will be displayed as the names of nodes in the enrichment map. The type of the gene set names should be one of the following: "id", "term" or "none".

displayEdgeLabel: a single logical value specifying whether or not to display the labels of the edges in the enrichment map

layout: a single character value specifying the layout of the enrichment map. (see `help(layout)` of the package `igraph`)

Details

The idea of this function is similar to the PLoS one paper by Merico et al.

An enrichment map is a network to help better visualize and interpret the GSEA or Hypergeometric test results. In an enrichment map, the nodes represent gene sets and the edges denote the Jaccard similarity coefficient between two gene sets. Node colors are scaled according to the adjusted p-values (the darker the more significant). For GSEA, nodes are colored by the sign of the enrichment scores (red:+, blue: -). The size of nodes illustrates the size of gene sets, while the width of edges denotes the Jaccard coefficient.

Value

an object of `igraph` with all attributes about the enrichement map

Author(s)

Xin Wang <xw264@cam.ac.uk>

References

Merico, D. et al, Enrichment Map: A Network-Based Method for Gene-Set Enrichment Visualization and Interpretation, PloS one, 2010, e13984

See Also

`plotEnrichMap`

Examples

```
## Not run:
library(org.Dm.eg.db)
library(KEGG.db)
##load data for enrichment analyses
data("KcViab_GSCA")
##plot and save the enrichment map
viewEnrichMap(KcViab_GSCA, gscs=c("GO_MF"), allSig=TRUE, ntop=NULL, gsNameType="id",
displayEdgeLabel=FALSE,layout="layout.fruchterman.reingold")
##append Gene set terms to results
KcViab_GSCA<-appendGSTerms(KcViab_GSCA, goGSCs=c("GO_BP","GO_MF","GO_CC"),
keggGSCs=c("PW_KEGG"))
viewEnrichMap(KcViab_GSCA, gscs=c("GO_MF"), allSig=TRUE, ntop=NULL, gsNameType="term",
displayEdgeLabel=FALSE,layout="layout.fruchterman.reingold")

## End(Not run)
```

---

viewGSEA *Plot a figure of GSEA results for one gene set*

Description

This is a generic function.

When implemented as the S4 method for objects of class `GSCA`, this function plots a figure of the positions of the gene sets in the ranked gene list and the location of the enrichment score.

To use this function for objects of class `GSCA`:

viewGSEA(object, gscName, gsName)

Usage

```
viewGSEA(object, ...)
```

Arguments

object	an object. When this function is implemented as the S4 method of class GSCA, this argument is an object of class GSCA.
...	other arguments. (see below for the arguments supported by class GSCA)
	gscName: a single character value specifying the name of the gene set collection where the gene set is
	gsName: a single character value specifying the name of the gene set to be plotted

Details

We suggest to print the names of top significant gene sets using the function getTopGeneSets before plotting the GSEA results.

Author(s)

Xin Wang <xw264@cam.ac.uk>

See Also

plotGSEA, gseaPlots

Examples

```
## Not run:
library(org.Dm.eg.db)
library(KEGG.db)
##load sample data
data("KcViab_GSCA")
##print summary of results
summarize(KcViab_GSCA, what="Result")
##print top significant gene sets in GO.BP
topPWKEGG<-getTopGeneSets(KcViab_GSCA, "GSEA.results", "PW_KEGG", allSig=TRUE)
##view a gene set
viewGSEA(KcViab_GSCA, "PW_KEGG", topPWKEGG[["PW_KEGG"]][1])

## End(Not run)
```

---

viewSubNet *Plot a figure of the enriched subnetwork*

Description

This is an generic function.

When implemented as the S4 method of class NWA, this function invokes the function networkPlot to plot the subnetwork identified by the 'BioNet' package.

Usage

```
viewSubNet(object)
```

Arguments

object	an object. When implemented as S4 methods of class NWA, this argument is an object of class 'GSCA'.

Details

After the analyses step for an object of class NWA, the user can generate the enriched subnetwork identified by the package 'BioNet'. If the slot 'phenotype' was inputted during the initialization of the object, this function will send it to the function networkPlot as the argument 'phenotypeVector' to highlight nodes in different colors. If the argument 'species' of the function analyze has been assigned, the labels of the nodes of this subnetwork will be mapped to gene symbols corresponding to the species; otherwise, the Entrez identifiers will be used as the labels.

Author(s)

Xin Wang <xw264@cam.ac.uk>

See Also

networkPlot, analyze, plotSubNet

Examples

```
## Not run:
##load sample data
data("KcViab_NWA")
##plot to screen the identified subnetwork
viewSubNet(KcViab_NWA)

## End(Not run)
```

---

writeReportHTSA *Write HTML reports for enrichment and/or network analyses*

Description

This function writes an HTML report following a complete analysis of a data set with the objects of class GSCA and/or NWA.

Usage

```
writeReportHTSA(gsca, nwa, experimentName="Unknown", species=NULL,
ntop=NULL, allSig=FALSE, keggGSCs=NULL, goGSCs=NULL, reportDir=
"HTSanalyzerReport")
```

Arguments

gsca	an object of class GSCA
nwa	an object of class NWA
experimentName	a single character value specifying the name of the experiment (just for you own record)

species	a single character value specifying the species for which the data should be read. The current version supports one of the following species: "Dm" ("Drosophila_melanogaster"), "Hs" ("Homo_sapiens"), "Rn" ("Rattus_ norvegicus"), "Mm" ("Mus_musculus"), "Ce" ("Caenorhabditis_elegans").
ntop	a single integer value specifying the number of plots to be produced for the GSEA analysis. For each gene set collection, plots are produced for the 'ntop' most significant gene sets.
allSig	a single logical value determining whether or not to generate plots for all significant gene sets. A gene set is significant if its corresponding adjusted p-value is less than the pValueCutoff set in function analyze.
keggGSCs	a character vector of the names of all KEGG gene set collections. This will help create web links for KEGG terms.
goGSCs	a character vector of the names of all GO gene set collections. This will help create web links for GO terms.
reportDir	a single character value specifying the directory to store reports

Details

This function takes in the objects of the two wrapper classes (GSCA and NWA) and writes a report into a user-specified directory. An index HTML file containing a summary of all results and hyperlinked tabs linking to more detailed results will be generated in the root directory. The other HTML files will be stored in a subdirectory called 'html'. All images including the GSEA plots, enrichment maps and subnetwork figure will be produced in a subdirectory called 'image'. All documents or text files such as the files containing significant gene sets of the hypergeometric test results will be stored in a subdirectory called 'doc'.

Author(s)

Xin Wang, Camille Terfve

See Also

report, reportAll

Examples

```
## Not run:
##(see the vignette for details about the preprocessing of this data set)
library(KEGG.db)
library(GO.db)
library(AnnotationDbi)
data("KcViab_GSCA")
data("KcViab_NWA")
##append gene set terms to results
KcViab_GSCA<-appendGSTerms(KcViab_GSCA, keggGSCs="PW_KEGG",
goGSCs=c("GO_BP","GO_MF","GO_CC"))
##report both analyses
writeReportHTSA(gsca=KcViab_GSCA, nwa=KcViab_NWA, experimentName="KcViab",
species="Dm", allSig=TRUE, keggGSCs="PW_KEGG", goGSCs=c("GO_BP","GO_MF",
"GO_CC"), reportDir="HTSanalyzerReport")
browseURL(file.path(getwd(), "HTSanalyzerReport", "index.html"))

## End(Not run)
```

Index