gage

April 20, 2011

`eg2sym` *Conversion between Entrez Gene IDs and official gene symbols for human genes.*

Description

These functions convert Entrez Gene IDs to official gene symbols for human genes, or vise versa.

Usage

```
eg2sym(eg)
sym2eg(sym)
```

Arguments

`eg`	character vector for Entrez Gene IDs (human genes only).
`sym`	character vector for official gene symbols (human genes only).

Details

Currently, only conversion for human genes are supported. Notice that some gene symbols are not official, hence not recognized and NA will be returned in such cases.

Value

A character vector giving the converted official gene symbols or Entrez IDs.

Author(s)

Weijun Luo <luo@cshl.edu>

References

Luo, W., Friedman, M., Shedden K., Hankenson, K. and Woolf, P GAGE: Generally Applicable Gene Set Enrichment for Pathways Analysis. BMC Bioinformatics 2009, 10:161

See Also

`egSymb` mapping data between Entrez Gene IDs and official symbols; `readList` read in gene set list

Examples

```
#genes in gse16873 was label by Entrez IDs
data(gse16873)
head(rownames(gse16873))
#may convert the gene IDs to official symbols
gse16873.sym<-gse16873
data(egSymb)
rownames(gse16873.sym)<-eg2sym(rownames(gse16873.sym))
head(rownames(gse16873.sym))

#convert kegg.gs correspondingly
data(kegg.gs)
kegg.gs.sym<-lapply(kegg.gs, eg2sym)
lapply(kegg.gs.sym[1:3],head)
#GAGE analysis with the converted data
cn=colnames(gse16873)
hn=grep('HN',cn, ignore.case =TRUE)
dcis=grep('DCIS',cn, ignore.case =TRUE)
gse16873.kegg.p2 <- gage(gse16873.sym, gsets = kegg.gs.sym,
    ref = hn, samp = dcis)
```

`egSymb` *Mapping between Entrez Gene IDs and official symbols*

Description

A two column matrix listing the Entrez IDs and official symbols for all currently known human genes.

Usage

```
data(egSymb)
```

Format

The format is: chr [1:40784, 1:2] "1" "10" "100" "1000" ... - attr(*, "dimnames")=List of 2 ..$: NULL ..$: chr [1:2] "eg" "sym"

Details

This mapping matrix is commonly used together with functions `eg2sym` and `sym2eg`. Check the examples below.

Source

This mapping data matrix were compiled using the gene data from NCBI Entrez Gene database.

Similar information can also be derived from Bioconductor package org.Hs.eg.db. Please check the package for more information.

References

Entrez Gene <URL: http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=gene>

Examples

```
#genes in gse16873 was label by Entrez IDs
data(gse16873)
head(rownames(gse16873))
#may convert the gene IDs to official symbols
gse16873.sym<-gse16873
data(egSymb)
rownames(gse16873.sym)<-eg2sym(rownames(gse16873.sym))
head(rownames(gse16873.sym))

#convert kegg.gs correspondingly
data(kegg.gs)
kegg.gs.sym<-lapply(kegg.gs, eg2sym)
lapply(kegg.gs.sym[1:3],head)
#GAGE analysis with the converted data
cn=colnames(gse16873)
hn=grep('HN',cn, ignore.case =TRUE)
dcis=grep('DCIS',cn, ignore.case =TRUE)
gse16873.kegg.p2 <- gage(gse16873.sym, gsets = kegg.gs.sym,
    ref = hn, samp = dcis)
```

---

esset.grp *The non-redundant signcant gene set list*

Description

This function extract a non-redundant signcant gene set list, groups of redundant gene sets, and related data from `gage` results. Redundant gene sets are those overlap heavily in their effective member gene lists or core genes.

Usage

```
esset.grp(setp, exprs, gsets, ref = NULL, samp = NULL, test4up = TRUE,
same.dir = TRUE, compare = "paired", use.fold = TRUE, cutoff = 0.01,
use.q = FALSE, pc = 10^-10, output = TRUE, outname = "esset.grp",
make.plot = FALSE, pdf.size = c(7, 7), core.counts = FALSE, get.essets =
TRUE, bins = 10, bsize = 1, cex = 0.5, layoutType = "circo", name.str =
c(10, 100), ...)
```

Arguments

`setp`	a numeric matrix, the result matrix returned by `gage` function. Check `gage` help information for details.
`exprs`	an expression matrix or matrix-like data structure, with genes as rows and samples as columns.
`gsets`	a named list, each element contains a gene set that is a character vector of gene IDs or symbols. For example, type `head(kegg.gs)`. A gene set can also be a "smc" object defined in PGSEA package. Make sure that the same gene ID system is used for both `gsets` and `exprs`.
`ref`	a numeric vector of column numbers for the reference condition or phenotype (i.e. the control group) in the exprs data matrix. Default ref = NULL, all columns are considered as target experiments.

samp	a numeric vector of column numbers for the target condition or phenotype (i.e. the experiment group) in the exprs data matrix. Default samp = NULL, all columns other than ref are considered as target experiments.
test4up	boolean, whether the input gage result or signficant gene sets are test results for up-regulated gene sets or not. This information is needed for selecting core member genes which contribute to the overall signficance of a gene sets.
same.dir	boolean, whether the input gage result test for changes in a gene set toward a single direction (all genes up or down regulated) or changes towards both directions simultaneously.
compare	character, which comparison scheme to be used: 'paired', 'unpaired', '1on-group', 'as.group'. 'paired' is the default, ref and samp are of equal length and one-on-one paired by the original experimental design; 'as.group', group-on-group comparison between ref and samp; 'unpaired' (used to be '1on1'), one-on-one comparison between all possible ref and samp combinations, although the original experimental design may not be one-on-one paired; '1on-group', comparison between one samp column at a time vs the average of all ref columns.
use.fold	Boolean, whether the input gage results used fold changes or t-test statistics as per gene statistics. Default use.fold= TRUE.
cutoff	numeric, p- or q-value cutoff, between 0 and 1. Default 0.01 (for p-value). When q-value is used, recommended cutoff value is 0.1.
use.q	boolean, whether to use q-value or not as the pre-selection of a signficant gene set list. Default to be FALSE, i.e. use the p-value instead.
pc	numeric, cutoff p-value for the overlap between gene sets to be called 'redundant', default to 10e-10, may need trial-and-error to find the best value.
output	boolean, whether output the non-redundant gene set list as tab-delimited text file? Default to be TRUE.
outname	character, the prefix used to label the output file names when output = TRUE.
make.plot	boolean, whether to generate the network graph to visualize the redundancy (overlap in core genes) between significant gene sets. Currently the only feasible option is FALSE.
pdf.size	numeric vector of length 2, spcifies the PDF file size for network graph outpout. Currently unsupported.
core.counts	Currently unsupported.
get.essets	Currently unsupported.
bins	Currently unsupported.
bsize	Currently unsupported.
cex	Currently unsupported.
layoutType	Currently unsupported.
name.str	numeric vector of length 2, specifies the substring range of the gene set name to show in the network graph. Currently unsupported.
...	extra arguments to be passed into internal function make.graph. Currently unsupported.

Details

Redundant gene sets are defined to be those overlap heavily in their effective member gene lists or core genes. Core genes are those member genes that really contribute to the signficance of the gene set in GAGE analysis in the interesting direction(s). Argument `pc` set the cutoff for the overlap to be called "redundant". The redundancy between gene sets is then represented by a undirected graph/network. Groups of redundant gene sets are then derived as the connected component in the network graph.

The selection criterion for gene sets here is p-value, instead of the commonly used q-value. This is because for extracting a non-redundant list of signficant gene sets, p-value is relative stable, but q-value changes when the total number of gene sets being considered changes. Of course, q-value is also a sensible selection criterion, when one take this step as a further refinement on the list of signficant gene sets.

Value

The value returned by `pairData` is a list of 7 elements:

`essentialSets`
: character vector, the non-redundant signficant gene set list.

`setGroups`
: list, each element is a character vector of a group of redundant gene sets.

`allSets`
: character vector, the full list of signficant gene sets.

`setGroups`
: list, each element is a character vector of a connected component in the redundancy graph representation of the gene set.

`overlapCounts`
: numeric matrix, the overlap core gene counts between the signficant gene sets.

`overlapPvals`
: numeric matrix, the significance (in p-values) of the overlap core gene counts between the signficant gene sets.

`coreGeneSets`
: list, each element is a character vector of the core genes in a significant gene set.

Author(s)

Weijun Luo <luo@cshl.edu>

References

Luo, W., Friedman, M., Shedden K., Hankenson, K. and Woolf, P GAGE: Generally Applicable Gene Set Enrichment for Pathways Analysis. BMC Bioinformatics 2009, 10:161

See Also

`gage` the main function for GAGE analysis; `sigGeneSet` significant gene set from GAGE analysis; `essGene` essential member genes in a gene set;

Examples

```
data(gse16873)
cn=colnames(gse16873)
hn=grep('HN',cn, ignore.case =TRUE)
dcis=grep('DCIS',cn, ignore.case =TRUE)
data(kegg.gs)
```

```
#kegg test for 1-directional changes
gse16873.kegg.p <- gage(gse16873, gsets = kegg.gs,
    ref = hn, samp = dcis)
#kegg test for 2-directional changes
gse16873.kegg.2d.p <- gage(gse16873, gsets = kegg.gs,
    ref = hn, samp = dcis, same.dir = FALSE)
gse16873.kegg.esg.up <- esset.grp(gse16873.kegg.p$greater,
    gse16873, gsets = kegg.gs, ref = hn, samp = dcis,
    test4up = TRUE, output = TRUE, outname = "gse16873.kegg.up", make.plot = FALSE)
gse16873.kegg.esg.dn <- esset.grp(gse16873.kegg.p$less,
    gse16873, gsets = kegg.gs, ref = hn, samp = dcis,
    test4up = FALSE, output = TRUE, outname = "gse16873.kegg.dn", make.plot = FALSE)
gse16873.kegg.esg.2d <- esset.grp(gse16873.kegg.2d.p,
    gse16873, gsets = kegg.gs, ref = hn, samp = dcis,
    test4up = TRUE, output = TRUE, outname = "gse16873.kegg.2d", make.plot = FALSE)
names(gse16873.kegg.esg.up)
head(gse16873.kegg.esg.up$essentialSets, 4)
head(gse16873.kegg.esg.up$setGroups, 4)
head(gse16873.kegg.esg.up$coreGeneSets, 4)
```

`essGene` *Essential member genes in a gene set*

Description

This function extracts data for essential member genes in a gene set. Essential genes are genes that have changes over noise level.

Usage

```
essGene(gs, exprs, ref = NULL, samp = NULL, gsets = NULL, compare
= "paired", use.fold = TRUE, rank.abs = FALSE, use.chi = FALSE, chi.p =
0.05, ...)
```

Arguments

`gs`	character, either the name of an interesting gene set in a gene set collection passed by `gsets` argument, or a vector of gene IDs. Make sure that the same gene ID system is used for both `gs` and `exprs`.
`exprs`	an expression matrix or matrix-like data structure, with genes as rows and samples as columns.
`ref`	a numeric vector of column numbers for the reference condition or phenotype (i.e. the control group) in the exprs data matrix. Default ref = NULL, all columns are considered as target experiments.
`samp`	a numeric vector of column numbers for the target condition or phenotype (i.e. the experiment group) in the exprs data matrix. Default samp = NULL, all columns other than ref are considered as target experiments.
`gsets`	a named list, each element contains a gene set that is a character vector of gene IDs or symbols. For example, type `head(kegg.gs)`. A gene set can also be a "smc" object defined in PGSEA package. Make sure that the same gene ID system is used for both `gsets` and `exprs`. Default to be NULL, then argument `gs` needs to be a vector of gene IDs.

compare	character, which comparison scheme to be used: 'paired', 'unpaired', '1on-group', 'as.group'. 'paired' is the default, ref and samp are of equal length and one-on-one paired by the original experimental design; 'as.group', group-on-group comparison between ref and samp; 'unpaired' (used to be '1on1'), one-on-one comparison between all possible ref and samp combinations, although the original experimental design may not be one-on-one paired; '1on-group', comparison between one samp column at a time vs the average of all ref columns.
use.fold	Boolean, whether the input gage results used fold changes or t-test statistics as per gene statistics. Default use.fold= TRUE.
rank.abs	boolean, whether to sort the essential gene data based on absoluate changes. Default to be FALSE.
use.chi	boolean, whether to use chi-square test to select the essential genes. Default to be FALSE, use the mean plus standard deviation of all gene changes instead. Check details for more information.
chi.p	numeric value between 0 and 1, cutoff p-value for the chi-square test to select the essential genes. Default to 0.05.
...	other arguments to be passed into the inside gagePrep function.

Details

There are two different criteria for essential gene selection. One uses a chi-square test to determin whether the change of a gene is more than noise. A second considers any changes beyond 1 standard deviation from mean of all genes as real.

Note that essential genes are different from core genes considered in esset.grp function. Essential genes may change in a different direction than the overall change of a gene set. But core genes need to change in the in the interesting direction(s) of the gene set test.

Value

A expression data matrix extracted for the essential genes, with similar structure as exprs.

Author(s)

Weijun Luo <luo@cshl.edu>

References

Luo, W., Friedman, M., Shedden K., Hankenson, K. and Woolf, P GAGE: Generally Applicable Gene Set Enrichment for Pathways Analysis. BMC Bioinformatics 2009, 10:161

See Also

gage the main function for GAGE analysis; geneData output and visualization of expression data for selected genes; esset.grp non-redundant signcant gene set list;

Examples

```
data(gse16873)
cn=colnames(gse16873)
hn=grep('HN',cn, ignore.case =TRUE)
dcis=grep('DCIS',cn, ignore.case =TRUE)
```

```
#kegg test for 1-directional changes
data(kegg.gs)
gse16873.kegg.p <- gage(gse16873, gsets = kegg.gs,
    ref = hn, samp = dcis)
rownames(gse16873.kegg.p$greater)[1:3]
gs=unique(unlist(kegg.gs[rownames(gse16873.kegg.p$greater)[1:3]]))
essData=essGene(gs, gse16873, ref =hn, samp =dcis)
head(essData)
ref1=1:6
samp1=7:12
#generated text file for data table, pdf files for heatmap and scatterplot
for (gs in rownames(gse16873.kegg.p$greater)[1:3]) {
    outname = gsub(" |:|/", "_", substr(gs, 10, 100))
    geneData(genes = kegg.gs[[gs]], exprs = essData, ref = ref1,
        samp = samp1, outname = outname, txt = TRUE, heatmap = TRUE,
        Colv = FALSE, Rowv = FALSE, dendrogram = "none", limit = 3, scatterplot = TRUE)
}
```

`gageComp` *Compare multiple GAGE analyses results*

Description

This function is used to compare the results after running multiple rounds of GAGE analysis. It is frequently used after batch analysis using `gagePipe`, but may also be used after multiple runs of `gage` manually.

Usage

```
gageComp(sampnames, dataname, gsname = c("kegg.gs", "go.gs"), use.cols =
c("stat.mean", "q.BH"), q.cutoff = 0.1, do.plot = TRUE)
```

Arguments

`sampnames`	character vector, the names of the sample groups, on which the GAGE analysis has been done and to be compared. This same argument is used in `gagePipe` function. These `sampnames` have been used to label `gage` result objects.
`dataname`	character, the name of the data on which the GAGE analysis has been done. This same argument is used in `gagePipe` function. This name has be included as the prefix of the GAGE analysis output file names, and will be used in the comparison output file names.
`gsname`	character, the name(s) of the gene set collection(s) to be considered in the comparison. In other words, this argument specifies GAGE analysis results with what type(s) of gene sets are to be compared on. Default to be `c("kegg.gs", "go.gs")`.
`use.cols`	character, what columns in the `gage` analysis result objects will be used in the comparison. Default to be "stat.mean" (mean of gene set test statistics) and "q.BH" (q-value using BH procedure). Check help information for `gage` function for more details on the result columns.
`q.cutoff`	numeric, q-value cutoff between 0 and 1 for signficant gene sets selection. Default to be 0.1. The same argument is used in `gagePipe` function.

do.plot boolean, whether to plot the venn diagram for the comparison results. Default to be TRUE.

Details

gageComp works with the results of gagePipe run by default. Try to load the .RData file named after dataname first. It there is no such file, it assumes that the gage result objects have been loaded and exist in the global environment.

For the GAGE analysis results with each gene set collection specified in gsname, gagePipe compares the signficant gene set lists between the sample groups specified in sampnames. For each gene set collection, three comparisons will be done, on the 2-direction perturbed, up-regulated, and down-regulated gene sets.

The comparison results are output as tab-delimited text files. Venn digrams are only plot for comparison between 2-3 parties. But the text file outputs are not limited by the number of parties under comparison. The venn diagram is generated by calling a revised function based on the VennDigram function from limma package.

Value

The function returns invisible 1 when successfully executed.

Author(s)

Weijun Luo <luo@cshl.edu>

References

Luo, W., Friedman, M., Shedden K., Hankenson, K. and Woolf, P GAGE: Generally Applicable Gene Set Enrichment for Pathways Analysis. BMC Bioinformatics 2009, 10:161

See Also

gagePipe pipeline for multiple GAGE analysis in a batch; gage the main function for GAGE analysis

Examples

```
data(gse16873)
cn=colnames(gse16873)
hn=grep('HN',cn, ignore.case =TRUE)
dcis=grep('DCIS',cn, ignore.case =TRUE)
data(kegg.gs)

library(gageData)
data(gse16873.2)
cn2=colnames(gse16873.2)
hn2=grep('HN',cn2, ignore.case =TRUE)
dcis2=grep('DCIS',cn2, ignore.case =TRUE)

#multiple GAGE analysis in a batch with the combined data
gse16873=cbind(gse16873, gse16873.2)
dataname='gse16873' #output data prefix
sampnames=c('dcis.1', 'dcis.2')
refList=list(hn, hn2+12)
sampList=list(dcis, dcis2+12)
```

```
gagePipe(gse16873, gsname = "kegg.gs", dataname = "gse16873",
    sampnames = sampnames, ref.list = refList, samp.list = sampList,
    comp.list = "paired")

#follow up comparison between the analyses
load('gse16873.gage.RData')
#list gage result objects
objects(pat = "[.]p$")
gageComp(sampnames, dataname, gsname = "kegg.gs",
    do.plot = TRUE)
```

`gagePipe` *GAGE analysis pipeline*

Description

Function gagePipe runs mutliple rounds of GAGE in a batch without interference, and outputs signcant gene set lists in text format and save the results in RData format.

Usage

```
gagePipe(arraydata, dataname = "arraydata", trim.at=TRUE, sampnames, gsdata = NU
gsname = c("kegg.gs", "go.gs"), ref.list, samp.list, weight.list = NULL,
comp.list = "paired", q.cutoff = 0.1, ...)
```

Arguments

`arraydata`	corresponds to `exprs` argument for `gage` function. But can either be a matrix-like data structure when the data has been loaded into R or the full path to the data file in .RData format if the data has not been loaded.
`dataname`	character, the name of the data to be analyzed. This name will be included as the prefix of the output file names. Default to be "arraydata".
`trim.at`	boolean, whether to trim the suffix "_at" from the probe set IDs or row names of the microarray data. Default to be TRUE.
`sampnames`	character vector, the names of the sample groups, on which the GAGE analysis is done. Each sample groups corresponds to one element of samp.list and the matching element of ref.list. These names will be included in the output file names or object names.
`gsdata`	character, the full path to the gene set data file in .RData format if the data has not been loaded. Default to be NULL, i.e. the gene set data has been loaded. Make sure that the same gene ID system is used for both `gsdata` and `arraydata`.
`gsname`	character, the name(s) of the gene set collections to be used. Default to be `c("kegg.gs", "go.gs")`.
`ref.list`	a list of `ref` inputs for `gage` function. In other words, each element of the list is a column number vector for the reference condition or phenotype (i.e. the control group) in the exprs data matrix.
`samp.list`	a list of `samp` inputs for `gage` function. In other words, each element of the list is a column number vector for the target condition or phenotype (i.e. the experiment group) in the exprs data matrix.

weight.list	a list or a vector of weights input(s) for gage function. As a list, the length of weight.list should equal to the length of ref.list and samp.list or 1. The weight.list vector or its element vectors of should match the elements of ref.list and samp.list in length or being NULL. When weight.list is a vector or length 1 list, all analyses will use the same weights setting.
comp.list	a list or a vector of compare input(s) for gage function. The length of the list or vector should equal to the length of ref.list and samp.list or 1. In the latter case, all analyses will use the same comparison scheme. The same as compare, the element value(s) in comp.list can be 'paired', 'unpaired', '1ongroup' or 'as.group'. Default to be 'paired'.
q.cutoff	numeric, q-value cutoff between 0 and 1 for signficant gene sets selection. Default to be 0.1.
...	other arguments to be passed into gage.

Details

gagePipe carries two rounds of GAGE analysis on each sample groups for each gene set collection specified in gsnames: one test for 1-direction changes (up- or down-regualted gene sets), one test for 2-direction changes (two-way perturbed gene sets). Correspondingly, the gage result matrix for the signficant gene sets are written into two tab-delimited text files, named after the dataname and sampnames. Note that the text file for 1-direction changes tests combines results for both up- and down-regulated gene sets. Meanwhile, the full gage analysis result objects (matrices or lists of matrices) are saved into a .RData file. The result objects are name after the sampnames and gsnames.

Value

The function returns invisible 1 when successfully executed.

Author(s)

Weijun Luo <luo@cshl.edu>

References

Luo, W., Friedman, M., Shedden K., Hankenson, K. and Woolf, P GAGE: Generally Applicable Gene Set Enrichment for Pathways Analysis. BMC Bioinformatics 2009, 10:161

See Also

gage the main function for GAGE analysis; heter.gage GAGE analysis for heterogeneous data

Examples

```
data(gse16873)
cn=colnames(gse16873)
hn=grep('HN',cn, ignore.case =TRUE)
dcis=grep('DCIS',cn, ignore.case =TRUE)
data(kegg.gs)

library(gageData)
data(gse16873.2)
cn2=colnames(gse16873.2)
```

```
hn2=grep('HN',cn2, ignore.case =TRUE)
dcis2=grep('DCIS',cn2, ignore.case =TRUE)

#multiple GAGE analysis in a batch with the combined data
gse16873=cbind(gse16873, gse16873.2)
dataname='gse16873' #output data prefix
sampnames=c('dcis.1', 'dcis.2')
refList=list(hn, hn2+12)
sampList=list(dcis, dcis2+12)
gagePipe(gse16873, gsname = "kegg.gs", dataname = "gse16873",
    sampnames = sampnames, ref.list = refList, samp.list = sampList,
    comp.list = "paired")

#follow up comparison between the analyses
load('gse16873.gage.RData')
#list gage result objects
objects(pat = "[.]p$")
gageComp(sampnames, dataname, gsname = "kegg.gs",
    do.plot = TRUE)
```

`gage` *GAGE (Generally Applicable Gene-set Enrichment) analysis*

Description

Run GAGE analysis to infer gene sets (or pathways, functional groups etc) that are signficantly perturbed relative to all genes considered. GAGE is generally applicable to essentially all microarray dta independent of data attributes including sample size, experimental layout, study design, and all types of heterogeneity in data generation.

gage is the main function; gagePrep is the functions for the initial data preparation, especially sample pairing; gageSum carries out the final meta-test summarization.

Usage

```
gage(exprs, gsets, ref = NULL, samp = NULL, set.size = c(10, 500),
same.dir = TRUE, compare = "paired", rank.test = FALSE, use.fold = TRUE,
FDR.adj = TRUE, weights = NULL, full.table = FALSE, saaPrep = gagePrep,
saaTest = gs.tTest, saaSum = gageSum, ...)

gagePrep(exprs, ref = NULL, samp = NULL, same.dir = TRUE, compare =
"paired", rank.test = FALSE, use.fold = TRUE, weights = NULL, full.table =
FALSE, ...)

gageSum(p.results, ref = NULL, mstat = NULL, setsizes = NULL, same.dir =
TRUE, compare = "paired", use.fold = TRUE, weights = NULL, full.table =
FALSE, ...)
```

Arguments

`exprs`	an expression matrix or matrix-like data structure, with genes as rows and samples as columns.

`gsets` a named list, each element contains a gene set that is a character vector of gene IDs or symbols. For example, type `head(kegg.gs)`. A gene set can also be a "smc" object defined in PGSEA package. Please make sure that the same gene ID system is used for both `gsets` and `exprs`.

`ref` a numeric vector of column numbers for the reference condition or phenotype (i.e. the control group) in the exprs data matrix. Default ref = NULL, all columns are considered as target experiments.

`samp` a numeric vector of column numbers for the target condition or phenotype (i.e. the experiment group) in the exprs data matrix. Default samp = NULL, all columns other than ref are considered as target experiments.

`set.size` gene set size (number of genes) range to be considered for enrichment test. Tests for too small or too big gene sets are not robust statistically or informative biologically. Default to be set.size = c(10, 500).

`same.dir` boolean, whether to test for changes in a gene set toward a single direction (all genes up or down regulated) or changes towards both directions simultaneously. For experimentally derived gene sets, GO term groups, etc, coregulation is commonly the case, hence same.dir = TRUE (default); In KEGG, BioCarta pathways, genes frequently are not coregulated, hence it could be informative to let same.dir = FALSE. Although same.dir = TRUE could also be interesting for pathways.

`compare` character, which comparison scheme to be used: 'paired', 'unpaired', '1on-group', 'as.group'. 'paired' is the default, ref and samp are of equal length and one-on-one paired by the original experimental design; 'as.group', group-on-group comparison between ref and samp; 'unpaired' (used to be '1on1'), one-on-one comparison between all possible ref and samp combinations, although the original experimental design may not be one-on-one paired; '1on-group', comparison between one samp column at a time vs the average of all ref columns.

For PAGE-like analysis, the default is compare='as.group', which is the only option provided in the original PAGE method. All other comparison schemas are set here for direct comparison to gage.

`rank.test` rank.test: Boolean, whether do the optional rank based two-sample t-test (equivalent to the non-parametric Wilcoxon Mann-Whitney test) instead of parametric two-sample t-test. Default rank.test = FALSE. This argument should be used with respect to argument saaTest.

`use.fold` Boolean, whether to use fold changes or t-test statistics as per gene statistics. Default use.fold= TRUE.

`FDR.adj` Boolean, whether to do adjust for multiple testing as to control FDR (False dicovery rate). Default to be TRUE.

`weights` a numeric vector to specify the weights assigned to pairs of ref-samp. This is needed for data with both technical replicates and biological replicates as to count for the different contributions from the two types of replicates. This argument is also useful in manually paring ref-samp for unpaired data, as in pairData function. function. Default to be NULL.

`full.table` Boolean, whether to output the full table of all individual p-values from the pairwise comparisons of ref and samp. This option is only effective for over 10 ref-samp pairs, all individual p-values will be output when the number is less than or equal than 10. Default to be FALSE.

saaPrep	function used for data preparation for single array based analysis, including sanity check, sample pairing, per gene statistics calculation etc. Default to be gagePrep, i.e. the default data preparation routine for gage analysis.
saaTest	function used for gene set tests for single array based analysis. Default to be gs.tTest, which features a two-sample t-test for differential expression of gene sets. Other options includes: gs.zTest, using one-sample z-test as in PAGE, or gs.KSTest, using the non-parametric Kolmogorov-Smirnov tests as in GSEA. The two non-default options should only be used when rank.test = FALSE.
saaSum	function used for summarization of the results from single array analysis (i.e. pairwise comparison between ref and samp). This function should include a meta-test for a global p-value or summary statistis and a FDR adjustment for multi-testing issue. Default to be gageSum, i.e. the default data summarization routine for gage analysis.
p.results	a numeric matrix of p-values for up-regulation (greater than) or down-regulation (less than) tests, i.e. the 2nd or 3rd list element returned by calling saaTest function. Gene sets are rows, samp-ref pairs are columns
mstat	vector of test statistics mean for individual gene sets, the 4th list element returned by calling saaTest function.
setsizes	vector of effective set size (number of genes) individual gene sets, the 5th list element returned by calling saaTest function.
...	other arguments to be passed into the optional functions for saaPrep, saaTest and saaSum.

Details

We proposed a single array analysis (i.e. the one-on-one comparison) approach with GAGE. Here we made single array analysis a general workflow for gene set analysis. Single array analysis has 4 major steps: Step 1 sample pairing, Step 2 per gene tests, Step 3 gene set tests and Step 4 meta-test summarization. Correspondingly, this new main function, gage, is divided into 3 relatively independent modules. Module 1 input preparation covers step 1-2 of single array analysis. Module 2 corresponds to step 3 gene set test, and module 3 to step 4 meta-test summarization. These 3 modules become 3 argument functions to gage, saaPrep, saaTest and saaSum. The modulization made gage open to customization or plug-in routines at each steps and fully realize the general applicability of single array analysis. More examples will be included in a second vignette to demonstrate the customization with these modules.

Value

The result returned by gage function is either a single data matrix (same.dir = FALSE, test for two-directional changes) or a named list of two data matrix (same.dir = TRUE, test for single-direction changes) for the results of up- ($greater) and down- ($less) regulated gene sets. Each data matrix here has gene sets as rows sorted by global p-values, with multiple statistics columns, including:

P.geomean	geometric mean of the individual p-values from multiple single array based gene set tests
stat.mean	mean of the individual statistics from multiple single array based gene set tests
P.erlang	gloal p-value or summary of the individual p-values from multiple single array based gene set tests. This is the default p-value being used.
q.BH	FDR q-value adjustment of the global p-value using the Benjamini & Hochberg procedure implemented in multtest package. This is the default q-value being used.

`set.size` the effective gene set size, i.e. the number of genes included in the gene set test

`other columns`
optional columns of the individual p-values from multiple single array based gene set tests

The result returned by `gagePrep` is a data matrix derived from `exprs`, but ready for column-wise gene est tests. In the matrix, genes are rows, and columns are the per gene test statistics from the ref-samp pairwise comparison.

The result returned by `gageSum` is almost identical to the results of `gage` function, except without the optional columns of individual p-values and without row sorting by global p-values.

Author(s)

Weijun Luo <luo@cshl.edu>

References

Luo, W., Friedman, M., Shedden K., Hankenson, K. and Woolf, P GAGE: Generally Applicable Gene Set Enrichment for Pathways Analysis. BMC Bioinformatics 2009, 10:161

More information at: http://sysbio.engin.umich.edu/~luow/downloads.php

See Also

`gs.tTest`, `gs.zTest`, and `gs.KSTest` functions used for gene set test; `gagePipe` and `heter.gage` function used for multiple GAGE analysis in a batch or combined GAGE analysis on heterogeneous data

Examples

```
data(gse16873)
cn=colnames(gse16873)
hn=grep('HN',cn, ignore.case =TRUE)
dcis=grep('DCIS',cn, ignore.case =TRUE)
data(kegg.gs)
data(go.gs)

#kegg test for 1-directional changes
gse16873.kegg.p <- gage(gse16873, gsets = kegg.gs,
    ref = hn, samp = dcis)
#go.gs with the first 1000 entries as a fast example.
gse16873.go.p <- gage(gse16873, gsets = go.gs,
    ref = hn, samp = dcis)
str(gse16873.kegg.p)
head(gse16873.kegg.p$greater[, 1:5])
head(gse16873.kegg.p$less[, 1:5])
#kegg test for 2-directional changes
gse16873.kegg.2d.p <- gage(gse16873, gsets = kegg.gs,
    ref = hn, samp = dcis, same.dir = FALSE)
head(gse16873.kegg.2d.p[, 1:5])

###alternative ways to do GAGE analysis###
#with unpaired samples
gse16873.kegg.unpaired.p <- gage(gse16873, gsets = kegg.gs,
    ref = hn, samp = dcis, compare = "unpaired")
```

```
#other options to tweak includes:
#saaTest, use.fold, rank.test, etc. Check arguments section above for
#details and the vignette for more examples.
```

`geneData` *View the expression data for selected genes*

Description

This function outputs and visualizes the expression data for seleted genes. Potential output files include: a tab-delimited text file, a heatmap in PDF format, and a scatter plot in PDF format.

Usage

```
geneData(genes, exprs, ref = NULL, samp = NULL, outname = "array",
txt = TRUE, heatmap = FALSE, scatterplot = FALSE, samp.mean = FALSE,
pdf.size = c(7, 7), cols = NULL, scale = "row", limit = NULL,
label.groups = TRUE, ...)
```

Arguments

`genes`	character, either a vector of interesting genes IDs or a 2-column matrix, where the first column specifies gene IDs used in `expData` while the second column gives another type of IDs to use for the output data files.
`exprs`	an expression matrix or matrix-like data structure, with genes as rows and samples as columns.
`ref`	a numeric vector of column numbers for the reference condition or phenotype (i.e. the control group) in the exprs data matrix. Default ref = NULL, all columns are considered as target experiments.
`samp`	a numeric vector of column numbers for the target condition or phenotype (i.e. the experiment group) in the exprs data matrix. Default samp = NULL, all columns other than ref are considered as target experiments.
`outname`	a character string, to be used as the prefix of the output data files. Default to be "array".
`txt`	boolean, whether to output the selected gene data as a tab-delimited text file. Default to be TRUE.
`heatmap`	boolean, whether to plot heatmap for the selected gene data as a PDF file. Default to be FALSE.
`scatterplot`	boolean, whether to make scatter plot for the selected gene data as a PDF file. Default to be FALSE.
`samp.mean`	boolean, whether to take the mean of gene data over the ref and samp group when making the scatter plot. Default to be FALSE, i.e. make scatter plots for the first two ref-samp pairs and label them differently on the same graph panel.
`pdf.size`	a numeric vector to specify the the width and height of PDF graphics region in inches. Default to be c(7, 7).
`cols`	a character vector to specify colors used for the heatmap image blocks. Default to be NULL, i.e. to generate a green-red spectrum based on the gene data automatically.

scale	character indicating if the values should be centered and scaled in either the row direction or the column direction, or none for the heatmap. The default is "row", other options include "column" and "none".
limit	numeric value to specify the maximal absolute value of gene data to visualize using the heatmap. Gene data beyong will be reset to equal this value. Default to NULL, i.e. plot all gene data values. This argument allows optimal differentiation between most gene data values when extremely positive/negative values exsit and squeeze the normal-value region. Recommend limit = 3 when the gene data is scaled by row.
label.groups	boolean, whether to label the two sample groups, i.e. ref and samp, differently using side color bars along the heatmap area. Default to be TRUE.
...	other arguments to be passed into the inside heatmap2 function.

Details

This function integrated three most common presentation methods for gene expression data: tab-delimited text file, heatmap and scatter plot. Heatmap is ideal for visualizing relative changes with gene-wise standardized (or row-scaled) data. The heatmap is generated by calling a improved version of the heatmap.2 function from gplots package. Scatter plot is ideal for visualizing the modest or small but consistent changes over a gene set between two states under comparison.

Although geneData is designed to be a standard-alone function, it is frequently used in tandem with essGene function to present the changes of the essential genes in signficant gene sets.

Value

The function returns invisible 1 when successfully executed.

Author(s)

Weijun Luo <luo@cshl.edu>

References

Luo, W., Friedman, M., Shedden K., Hankenson, K. and Woolf, P GAGE: Generally Applicable Gene Set Enrichment for Pathways Analysis. BMC Bioinformatics 2009, 10:161

See Also

essGene extract the essential member genes in a gene set; gage the main function for GAGE analysis;

Examples

```
data(gse16873)
cn=colnames(gse16873)
hn=grep('HN',cn, ignore.case =TRUE)
dcis=grep('DCIS',cn, ignore.case =TRUE)

#kegg test for 1-directional changes
data(kegg.gs)
gse16873.kegg.p <- gage(gse16873, gsets = kegg.gs,
    ref = hn, samp = dcis)
rownames(gse16873.kegg.p$greater)[1:3]
```

```
gs=unique(unlist(kegg.gs[rownames(gse16873.kegg.p$greater)[1:3]]))
essData=essGene(gs, gse16873, ref =hn, samp =dcis)
head(essData)
ref1=1:6
samp1=7:12
#generated text file for data table, pdf files for heatmap and scatterplot
for (gs in rownames(gse16873.kegg.p$greater)[1:3]) {
    outname = gsub(" |:|/", "_", substr(gs, 10, 100))
    geneData(genes = kegg.gs[[gs]], exprs = essData, ref = ref1,
        samp = samp1, outname = outname, txt = TRUE, heatmap = TRUE,
        Colv = FALSE, Rowv = FALSE, dendrogram = "none", limit = 3, scatterplot = TRUE)
}
```

`gse16873` *GSE16873: a breast cancer microarray dataset*

Description

GSE16873 is a breast cancer study (Emery et al, 2009) downloaded from Gene Expression Omnibus (GEO). GSE16873 covers twelve patient cases, each with HN (histologically normal), ADH (ductal hyperplasia), and DCIS (ductal carcinoma in situ) RMA samples. Due to the size limit of this package, we split this GSE16873 into two halves, each including 6 patients with their HN and DCIS but not ADH tissue types. This gage package only includes the first half dataset for 6 patients as this example dataset gse16873. Most of our demo analyses are done on the first half dataset, except for the advanced analysis where we use both halves datasets with all 12 patients. Details section below gives more information.

Usage

```
data(gse16873)
```

Details

Raw data for these two half datasets were processed separately using two different methods, FARMS and RMA, respectively to generate the non-biological data heterogeneity. The first half dataset is named as gse16873, the second half dataset named gse16873.2. We also have the full dataset, gse16873.full, which includes all HN, ADH and DCIS samples of all 12 patients, processed together using FARMS. The second half dataset plus the full dataset and the original BMP6 dataset used in GAGE paper and earlier versions of gage is supplied with another package, gageData.

Source

GEO Dataset GSE16873: <URL: http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE16873>

References

Emery LA, Tripathi A, King C, Kavanah M, Mendez J, Stone MD, de las Morenas A, Sebastiani P, Rosenberg CL: Early dysregulation of cell adhesion and extracellular matrix pathways in breast cancer progression. Am J Pathol 2009, 175:1292-302.

Examples

```
data(gse16873)
#check the heterogenity of the two half datasets
boxplot(data.frame(gse16873))

#column/smaple names
cn=colnames(gse16873)
hn=grep('HN',cn, ignore.case =TRUE)
adh=grep('ADH',cn, ignore.case =TRUE)
dcis=grep('DCIS',cn, ignore.case =TRUE)
print(hn)
print(dcis)

data(kegg.gs)
lapply(kegg.gs[1:3],head)
head(rownames(gse16873))
gse16873.kegg.p <- gage(gse16873, gsets = kegg.gs,
    ref = hn, samp = dcis)
```

`gs.tTest` *Gene set differential expression test*

Description

These functions test for perturbation of gene sets relative to all genes in the microarray data. They are the testing module for `gage` and single array analysis workflow.

They use different statistical tests: gs.tTest uses two-sample t-test, gs.zTest uses one-sample z-test, gs,KSTest uses Kolmogorov-Smirnov test.

Usage

```
gs.tTest(exprs, gsets, set.size = c(10, 500), same.dir = TRUE, ...)
gs.zTest(exprs, gsets, set.size = c(10, 500), same.dir = TRUE, ...)
gs.KSTest(exprs, gsets, set.size = c(10, 500), same.dir = TRUE, ...)
```

Arguments

`exprs`	an expression matrix or matrix-like data structure, with genes as rows and samples as columns.
`gsets`	a named list, each element contains a gene set that is a character vector of gene IDs or symbols. For example, type head(kegg.gs). A gene set can also be a "smc" object defined in PGSEA package. Make sure that the same gene ID system is used for both `gsets` and `exprs`.
`set.size`	gene set size (number of genes) range to be considered for enrichment test. Tests for too small or too big gene sets are not robust statistically or informative biologically. Default to be set.size = c(10, 500).
`same.dir`	whether to test for changes in a gene set toward a single direction (all genes up or down regulated) or changes towards both directions simultaneously. For experimentally derived gene sets, GO term groups, etc, coregulation is commonly the case, hence same.dir = TRUE (default); In KEGG, BioCarta pathways, genes frequently are not coregulated, hence it could be informative to let same.dir = FALSE. Although same.dir = TRUE could also be interesting for pathways.

...	other arguments to be passed into the secondary functions, not used currently.

Details

These functions are the gene set test module for `gage` and single array analysis workflow. When used in `gage` function, the function names are optional values for `saaTest` argument. Check help information for `gage` for details.

These functions may also used independently without calling `gage` function.

Value

As the raw results of gene set tests, a list of 5 components is returned:

`results`	matrix of test statistics, gene sets are rows, samp-ref pairs are columns
`p.results`	matrix of p-values for up-regulation (greater than) tests, gene sets are rows, samp-ref pairs are columns
`ps.results`	matrix of p-values for down-regulation (less than) tests, gene sets are rows, samp-ref pairs are columns
`mstat`	vector of test statistics mean for individual gene sets
`setsizes`	vector of effective set size (number of genes) individual gene sets

Author(s)

Weijun Luo <luo@cshl.edu>

References

Luo, W., Friedman, M., Shedden K., Hankenson, K. and Woolf, P GAGE: Generally Applicable Gene Set Enrichment for Pathways Analysis. BMC Bioinformatics 2009, 10:161

See Also

`gage` the main function for GAGE analysis

Examples

```
data(gse16873)
cn=colnames(gse16873)
hn=grep('HN',cn, ignore.case =TRUE)
dcis=grep('DCIS',cn, ignore.case =TRUE)
data(kegg.gs)

#kegg test
exprs.gage = gagePrep(gse16873, ref = hn, samp = dcis)
str(exprs.gage)
rawRes = gs.tTest(exprs.gage, gsets = kegg.gs)
str(rawRes)
head(rawRes$results)
head(rawRes$p.results)
```

heter.gage *GAGE analysis for heterogeneous data*

Description

heter.gage is a wrapper function of gage for heterogeneous data. pairData prepares the heterogeneous data and related arguments for GAGE analysis.

Usage

```
heter.gage(exprs, gsets, ref.list, samp.list, comp.list = "paired",
use.fold = TRUE, ...)

pairData(exprs, ref.list, samp.list, comp.list = "paired", use.fold =
TRUE, ...)
```

Arguments

exprs	an expression matrix or matrix-like data structure, with genes as rows and samples as columns.
gsets	a named list, each element contains a gene set that is a character vector of gene IDs or symbols. For example, type head(kegg.gs). A gene set can also be a "smc" object defined in PGSEA package. Make sure that the same gene ID system is used for both gsets and exprs.
ref.list	a list of ref inputs for gage function. In other words, each element of the list is a column number vector for the reference condition or phenotype (i.e. the control group) in the exprs data matrix.
samp.list	a list of samp inputs for gage function. In other words, each element of the list is a column number vector for the target condition or phenotype (i.e. the experiment group) in the exprs data matrix.
comp.list	a list or a vector of compare input(s) for gage function. The length of the list or vector should equal to the length of ref.list and samp.list or 1. In the latter case, all analyses will use the same comparison scheme. The same as compare, the element value(s) in comp.list can be 'paired', 'unpaired', '1ongroup' or 'as.group'. Default to be 'paired'.
use.fold	Boolean, whether to use fold changes or t-test statistics as per gene statistics. Default use.fold= TRUE.
...	other arguments to be passed into gage.

Details

comp.list can be a list or vector of mixture values of 'paired' and 'unpaired' matching the experiment layouts of the heterogeneous data. In such cases, each ref-samp pairs and corresponding columns in the result data matrix after calling pairData are assigned different weights when calling gage in the next step. The inclusion of '1ongroup' and 'as.group' in comp.list would make weight assignment very complicated especially when the sample sizes are different for the individual experiments of the heterogeneous data.

Value

The output of `pairData` is a list of 2 elements:

`exprs`	a data matrix derived from the input expression data matrix `exprs`, but ready for column-wise gene est tests. In the matrix, genes are rows, and columns are the per gene test statistics from the ref-samp pairwise comparison.
`weights`	weights assigned to columns of the output data matrix `exprs` when calling `gage` next. The value may be NULL if `comp.list` are all 'paired'.

The result returned by `heter.gage` function is the same as result of `gage`, i.e. either a single data matrix (same.dir = FALSE, test for two-directional changes) or a named list of two data matrix (same.dir = TRUE, test for single-direction changes) for the results of up- ($greater) and down- ($less) regulated gene sets. Check help information for `gage` for details.

Author(s)

Weijun Luo <luo@cshl.edu>

References

Luo, W., Friedman, M., Shedden K., Hankenson, K. and Woolf, P GAGE: Generally Applicable Gene Set Enrichment for Pathways Analysis. BMC Bioinformatics 2009, 10:161

See Also

`gage` the main function for GAGE analysis; `gagePipe` pipeline for multiple GAGE analysis in a batch

Examples

```
data(gse16873)
cn=colnames(gse16873)
hn=grep('HN',cn, ignore.case =TRUE)
dcis=grep('DCIS',cn, ignore.case =TRUE)
data(kegg.gs)

library(gageData)
data(gse16873.2)
cn2=colnames(gse16873.2)
hn2=grep('HN',cn2, ignore.case =TRUE)
dcis2=grep('DCIS',cn2, ignore.case =TRUE)

#combined the two half dataset
gse16873=cbind(gse16873, gse16873.2)
refList=list(hn, hn2+12)
sampList=list(dcis, dcis2+12)

#quick look at the heterogeneity of the combined data
summary(gse16873[,hn[c(1:2,7:8)]])
#if graphic devices open:
#boxplot(data.frame(gse16873))
gse16873.kegg.heter.p <- heter.gage(gse16873, gsets = kegg.gs,
    ref.list = refList, samp.list = sampList)
gse16873.kegg.heter.2d.p <- heter.gage(gse16873, gsets = kegg.gs,
    ref.list = refList, samp.list = sampList, same.dir = FALSE)
```

```
str(gse16873.kegg.heter.p)
head(gse16873.kegg.heter.p$greater[, 1:5])
```

---

kegg.gs *Common gene set data collections*

Description

The gene set data collections derived from KEGG, GO and BioCarta databases.

Usage

```
data(kegg.gs)
data(go.gs)
data(carta.gs)
```

Format

kegg.gs is a named list of 205 elements. Each element is a character vector of member gene Entrez IDs for a single KEGG pathway. Type `head(kegg.gs, 3)` for the first 3 gene sets or pathways.

go.gs is a named list of 1000 elements in this package. It is a truncated list in this package. The ful list of go.gs has ~10000 elements and is provided with an experimental data package gageData. Each element is a character vector of member gene Entrez IDs for a single Gene Ontology term. Type `head(go.gs, 3)` for the first 3 gene sets or GO terms.

carta.gs is a named list of 259 elements. Each element is a character vector of member gene Entrez IDs for a single BioCarta pathway. Type `head(carta.gs, 3)` for the first 3 gene sets or pathways.

Details

These gene set data were compiled using Entrez Gene IDs, gene set names and mapping information from multiple Bioconductor packages, including: org.Hs.eg.db, kegg.db, go.db and cMAP. Please check the corresponding packages for more information.

We only provide gene set data for human with this package. For other species, please check the experiment data package list of Bioconductor website or use the Bioconductor package GSEABase to build the users' own gene set collections.

Source

Data from multiple Bioconductor packages, including: org.Hs.eg.db, kegg.db, go.db and cMAP.

References

Entrez Gene <URL: http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=gene> KEGG pathways <URL: ftp://ftp.genome.ad.jp/pub/kegg/pathways> Gene Ontology <URL: http://www.geneontology.org/> cMAP <URL: http://cmap.nci.nih.gov/PW>

Examples

```
#load expression and gene set data
data(gse16873)
cn=colnames(gse16873)
hn=grep('HN',cn, ignore.case =TRUE)
dcis=grep('DCIS',cn, ignore.case =TRUE)

data(kegg.gs)
data(go.gs)

#make sure the gene IDs are the same for expression data and gene set
#data
lapply(kegg.gs[1:3],head)
lapply(go.gs[1:3],head)
head(rownames(gse16873))

#GAGE analysis
gse16873.kegg.p <- gage(gse16873, gsets = kegg.gs,
    ref = hn, samp = dcis)
gse16873.go.p <- gage(gse16873, gsets = go.gs,
    ref = hn, samp = dcis)
```

---

readExpData *Read in expression data*

Description

This is a wrapper function of `read.delim` for reading in expression data matrix in tab-delimited format.

Usage

```
readExpData(file = "arrayData.txt", ...)
```

Arguments

`file`	character string, the full path name to the expression data file in tab-delimited format. Rows are genes, columns are array samples.
`...`	other arguments to be passed into `read.delim` function.

Details

`readExpData` is a wrapper function of `read.delim`. Please check help information of `read.delim` for more details.

Value

A data.frame (matrix-like) of gene expression data. Rows are genes, columns are array samples.

Author(s)

Weijun Luo <luo@cshl.edu>

References

Luo, W., Friedman, M., Shedden K., Hankenson, K. and Woolf, P GAGE: Generally Applicable Gene Set Enrichment for Pathways Analysis. BMC Bioinformatics 2009, 10:161

See Also

readList read in gene set list

Examples

```
filename=system.file("data/gse16873.demo", package = "gage")
demo.data=readExpData(filename, row.names=1)
head(demo.data)
```

---

readList *Read in gene set data as a named list*

Description

This function reads in gene set data in GMT (.gmt) format as a named list. GMT is defined originally by GSEA program. The code may be slightly revised for reading in gene set data in other tab-delimited formats too.

Usage

```
readList(file)
```

Arguments

file	character string, the full path name to the gene set data file in GMT format.

Value

A named list, each element is a character vector giving the gene IDs of a gene set.

Author(s)

Weijun Luo <luo@cshl.edu>

References

Luo, W., Friedman, M., Shedden K., Hankenson, K. and Woolf, P GAGE: Generally Applicable Gene Set Enrichment for Pathways Analysis. BMC Bioinformatics 2009, 10:161

See Also

readExpData read in gene expression data

Examples

```
#an example GMT gene set data derived from MSigDB data
filename=system.file("data/c2.demo.gmt", package = "gage")
demo.gs=readList(filename)
demo.gs[1:3]
#to use these gene sets with gse16873, need to convert the gene symbols
#to Entrez IDs first
data(egSymb)
demo.gs.sym<-lapply(demo.gs, sym2eg)
demo.gs.sym[1:3]
```

`sigGeneSet` *Significant gene set from GAGE analysis*

Description

This function sorts and counts signcant gene sets based on q- or p-value cutoff.

Usage

```
sigGeneSet(setp, cutoff = 0.1, dualSig = (0:2)[2], qpval = c("q.BH",
"P.erlang")[1])
```

Arguments

`setp`	the result object returned by `gage` function, either a numeric matrix or a list of two such matrices. Check `gage` help information for details.
`cutoff`	numeric, q- or p-value cutoff, between 0 and 1. Default 0.1 (for q-value). When p-value is used, recommended cutoff value is 0.001 for data with more than 2 replicates per condition or 0.01 for les sample sizes.
`dualSig`	integer, switch argument controlling how dual-signficant gene sets should be treated. 0: discard such gene sets from the final significant gene set list; 1: keep such gene sets in the more significant direction and remove them from the less significant direction; 2: keep such gene sets in the lists for both directions. default to 1. Dual-signficant means a gene set is called significant simultaneously in both 1-direction tests (up- and down-regulated). Check the details for more information.
`qpval`	character, specifies the column name used for gene set selection, i.e. what type of q- or p-value to use in gene set selection. Default to be "q.BH" (q-value using BH procedure). "P.erlang" is the unadjusted global p-value and may be used as selection criterion sometimes.

Details

Dual signficant gene sets are rare cases, but do occurred in large clinic datasets. Gene sets are signficantly up-regulated in a subset of experiments, but down-regulated in another subset. Note that dual-signficant gene sets are not the same as gene sets called signficant in 2-directional tests, although they are related.

Value

`sigGeneSet` function returns a data matrix of the same structure as `gage` result matrix. Check `gage` help information for details. This data matrix combined both up-regulated and down-regulated gene set data for 1-directional tests.

Author(s)

Weijun Luo <luo@cshl.edu>

References

Luo, W., Friedman, M., Shedden K., Hankenson, K. and Woolf, P GAGE: Generally Applicable Gene Set Enrichment for Pathways Analysis. BMC Bioinformatics 2009, 10:161

See Also

`gage` the main function for GAGE analysis; `esset.grp` non-redundant signcant gene set list; `essGene` essential member genes in a gene set;

Examples

```
data(gse16873)
cn=colnames(gse16873)
hn=grep('HN',cn, ignore.case =TRUE)
dcis=grep('DCIS',cn, ignore.case =TRUE)
data(kegg.gs)

#kegg test for 1-directional changes
gse16873.kegg.p <- gage(gse16873, gsets = kegg.gs,
    ref = hn, samp = dcis)
#kegg test for 2-directional changes
gse16873.kegg.2d.p <- gage(gse16873, gsets = kegg.gs,
    ref = hn, samp = dcis, same.dir = FALSE)
gse16873.kegg.sig<-sigGeneSet(gse16873.kegg.p, dualSig=1)
str(gse16873.kegg.sig)
gse16873.kegg.2d.sig<-sigGeneSet(gse16873.kegg.2d.p)
str(gse16873.kegg.2d.sig)
```

Index