Package 'NEArender'

March 21, 2018

Title Network Enrichment Analysis

Version 1.5

Maintainer Ashwini Jeggari <ashwinipriya.jeggari@ki.se>

Description Performs network enrichment analysis against functional gene sets. Benchmarks networks. Renders raw gene profile matrices of dimensionality ``N genes x N samples" into the space of gene set (typically pathway) enrichment scores of dimensionality ``N pathways x N samples".

Depends R (>= 3.0.0)

License GPL-3

Encoding UTF-8

LazyData true

Imports stats, utils, ROCR, graphics, parallel, hexbin, MASS, RColorBrewer

Suggests knitr, rmarkdown, tinytex, data.table

RoxygenNote 6.0.1.9000

VignetteBuilder knitr

NeedsCompilation no

Author Andrey Alexeyenko [aut, cph, ths], Ashwini Jeggari [aut, cre]

Repository CRAN

Date/Publication 2018-03-21 19:44:55 UTC

R topics documented:

`as_genes_fgs` *Create single-gene FGS*

Description

Each network node X becomes an FGS itself, so that e.g. each ID 'X' generates a list entry named 'X', the content of which is c('X'), i.e. this same gene/protein. The total length of the FGS list then equals the number of nodes in the network. This is a way to create single-gene "quasi-pathways" for a more specific network enrichment analysis.

Usage

```
as_genes_fgs(Net.list, Lowercase = 1)
```

Arguments

`Net.list`	the global network object, pre-created using `import.net`
`Lowercase`	render gene/protein IDs lower-case (Default:1)

See Also

`import.net`, `import.gs`

Examples

```
data(net.kegg)
net <- import.net(net.kegg)
fgs.genes <- as_genes_fgs(net)
print(fgs.genes[1:10])
```

benchmark *Benchmark networks using Network Enrichment Analysis (NEA)*

Description

Tests the ability of a given network to perform well in a network enrichment analysis. It executes a series of multiple individual tests: for each member gene of a pathway or another functional set calculates the network enrichment score against other members of the same gene set. This procedure gives true positive and false negative test results. In order to complement it with false positives and true negatives, the same is done for randomly picked genes (with matching node connectivity values) against the same functional sets. The two vectors allow plotting a ROC curve where at each sequential cutoff represents a ratio of true positive vs. false positive predictions. This approach (first presented in Merid et al. (2012)) is an alternative to the trivial counting edges shared between different networks and is superior to the latter because: 1) the analysis can be done without knowing the "true" reference network, 2) benchmarks can be context-dependent by using domain-specific test sets (e.g. cancer, diabetes etc.), 3) one can compare more than two networks at a time, and 4) given dense global networks and due to the use of multi-gene sets, presence or absence of particular links is unlikely to affect the overall result.

Usage

```
benchmark(NET, GS, gs.gene.col = 2, gs.group.col = 3, net.gene1.col = 1,
  net.gene2.col = 2, echo = 1, graph = FALSE, na.replace = 0,
  mask = ".", minN = 0, coff.z = 1.965, coff.fdr = 0.1,
  Parallelize = 1)
```

Arguments

NET	A network to benchmark. See Details in nea.render.
GS	a test set, typically a set of pathways with known members.
gs.gene.col	number of the column containing GS genes (only needed if GS is submitted as a text file)
gs.group.col	number of the column containing group IDs (only needed if GS is submitted as a text file)
net.gene1.col	number of the column containing first nodes of each network edge (only needed if NET is submitted as a text file)
net.gene2.col	number of the column containing second nodes of each network edge (only needed if NET is submitted as a text file)
echo	if messages about execution progress should appear
graph	Plot the ROC curve immediately. Alternatively, the returned list is plotted afterwards by roc. In the latter case, it could be a combined list of lists for multiple test sets and networks which are then plotted as separate curves (see Examples).
na.replace	replace NA values. Default=0, i.e. do not replace.

mask	when the test set contains various GSs, they can be used selectively by applying a mask. The mask follows the regular expression synthax, since fixed=FALSE in grep.
minN	the minimal number of network edges that must connect a tested member with the GS genes for the test to be considered positive. (Default:0).
coff.z	a parameter to roc.
coff.fdr	to make significance levels comparable between different curves, the point where FDR=coff.fdr will be labeled with a circle (think of TP/FP ratio at this level).
Parallelize	The number of CPU cores to be used for the step "Counting actual links" (while the other steps are sufficiently fast). The option is not supported in Windows.

Details

The function would either plot a ROC curve for the analyzed network, or return an object with the following slots from function prediction (package ROCR):
tp, vector of true positives;
fp, vector of false positives;
tn, vector of true negatives;
fn, vector of false negatives;
cutoffs, z-score cutoffs from nea.render;
cross.z, a z-score value which corresponds to FDR=coff.fdr (will be denoted with a special marker at the curve);

Value

An object, i.e. a list of three equal-length vectors from a prediction object of ROCR package (prediction@cutoffs, prediction@fp, prediction@tp) and the point that matches coff.fdr. These are needed to plot a ROC curve for the given network and test set by using roc.

References

http://www.biomedcentral.com/1471-2105/15/308

See Also

roc, nea.render

Examples

```
data(can.sig.go);
fpath <- can.sig.go
gs.list <- import.gs(fpath, Lowercase = 1, col.gene = 2, col.set = 3);
data(net.kegg)
netpath <- net.kegg
net <- import.net(netpath)

b0 <- benchmark (NET = net,
 GS = gs.list,
 echo=1, graph=TRUE, na.replace = 0, mask = ".", minN = 0,
```

```
 coff.z = 1.965, coff.fdr = 0.1, Parallelize=2);

## Not run:
## Benchmark a number of networks on GO terms and KEGG pathways separately, using masks:
b1 <- NULL;
for (mask in c("kegg_", "go_")) {
b1[[mask]] <- NULL;
for (file.net in c("netpath")) {
# a series of networks can be put here: c("netpath1", "netpath2", "netpath3")
net <- import.net(netpath, col.1 = 1, col.2 = 2, Lowercase = 1, echo = 1)
b1[[mask]][[file.net]] <- benchmark (NET = net, GS = gs.list, echo=1,
graph=FALSE, na.replace = 0, mask = mask, minN = 0,  Parallelize=1);
}}
par(mfrow=c(2,1));
roc(b1[["kegg_"]], coff.z = 2.57,main="kegg_");
roc(b1[["go_"]], coff.z = 2.57,main="go_");

## End(Not run)
```

---

can.sig.go *Example functional gene sets (FGS) from GO and KEGG*

Description

Example functional gene sets (FGS) from GO and KEGG

Usage

```
data(can.sig.go)
```

Format

An object, i.e. a list with FGS IDs as names and vectors of gene IDs as member (34 FGSs in total). See see `import.gs`

Examples

```
head(can.sig.go)
```

connectivity *Connectivity*

Description

Function for plotting node degree distribution in order to test if the network is scale-free

Usage

```
connectivity(NET, Lowercase = 1, col.1 = 1, col.2 = 2, col.score = 3,
  echo = 1, main = "Connectivity plot", min.score = NA, n.top = NA,
  hex = FALSE)
```

Arguments

NET	Input network file.
Lowercase	If node IDs should be rendered lower-case (Default:1, i.e. 'yes').
col.1	Number of column where 1st node of each edge should be found (only needed when NET is a text file rather than a list).
col.2	Number of column where 2nd node of each edge should be found (only needed when NET is a text file rather than a list, i.e. passed down to import.net).
col.score	Number of column where edge confidence score is found (only needed when NET is a text file rather than a list, i.e. passed down to import.net).
echo	If messages about execution progress should appear.
main	title name for the plot, default: "Connectivity plot"
min.score	Minimum confidence score for an edge to be included in the network (is alternative to n.top and is only used when NET is a text file rather than a list, i.e. passed down to import.net).
n.top	Number of edges to be included in the network, top when ranked by confidence score (is alternative to min.score and only used when NET is a text file rather than a list, i.e. passed down to import.net).
hex	If the node degree distribution should be presented as density plot using package hexbin.

See Also

benchmark and import.net

Examples

```
file <-  system.file("extdata", "Connectivity.FC1_full", package = "NEArender")

connect <- connectivity(file,hex=TRUE)
```

fantom5.43samples *A Fantom5 transcriptomes in 43 carincinoma cell samples*

Description

A Fantom5 transcriptomes in 43 carincinoma cell samples

Usage

```
data(fantom5.43samples)
```

Format

An expression data matrix with the genes as rows and columns as cell line samples

Examples

```
colnames(fantom5.43samples)
```

gsea.render *Gene Set Enrichment Analysis (GSEA)*

Description

A binomial version of GSEA, unified as much as possible with `nea.render`. Given the altered gene sets (AGS) and functional gene sets (FGS), calculates no. of members (genes/protein IDs) shared by each AGS-FGS pair as well as respective enrichment statistics. Returns matrices of size *length(FGS) x length(AGS)* (see "Value"). Each of these two parameters can be submitted as either a text file or as an R list which have been preloaded with `import.gs`.

Usage

```
gsea.render(AGS, FGS, Lowercase = 1, ags.gene.col = 2, ags.group.col = 3,
  fgs.gene.col = 2, fgs.group.col = 3, echo = 1, Ntotal = 20000,
  Parallelize = 1)
```

Arguments

AGS	Either a text file or a list of members of each AGS (see Details). Group IDs should be found in `ags.group.col` and gene IDs would be found in `ags.gene.col`. Identical to AGS needed for in `nea.render` - see also details there.
FGS	Either a text file or a list of members of each FGS (see Details). Group IDs should be found in `fgs.group.col` and gene IDs would be found in `fgs.gene.col`. Alsmost identical to FGS needed for in `nea.render`.

Lowercase	render node and group IDs lower-case (Default:1, i.e. 'yes').
ags.gene.col	number of the column containing AGS genes (only needed if AGS is submitted as a text file).
ags.group.col	number of the column containing group IDs (only needed if AGS is submitted as a text file).
fgs.gene.col	number of the column containing FGS genes (only needed if FGS is submitted as a text file).
fgs.group.col	number of the column containing group IDs (only needed if FGS is submitted as a text file).
echo	if messages about execution progress should appear.
Ntotal	The important parameter for precise calculation of the Fisher's statistics: how big is the whole gene universe? Defaults to 20000 but should be changed depending on the hypothesis and genome/proteome size.
Parallelize	The number of CPU cores to be used for calculating the gene set overlap. The other steps are sufficiently fast. The option is not supported in Windows.

Value

A list of entries estimate, p, q, and n, each of which is a matrix of size length(FGS) x length(AGS). The two former ones contain respective output of fisher.test: p.value and estimate, whereas q is produced by p.adjust(p.value, method="BH") and n is the no. of shared members. Input to fisher.test is matrix(c(<no. of shared members>, <no. of solely FGS members>, <no. of solely AGS memb

See Also

nea.render, import.gs

Examples

```
ags.list <- samples2ags(fantom5.43samples, Ntop=20, method="topnorm")
data(can.sig.go)
fpath <- can.sig.go
fgs.list <- import.gs(fpath)
g1 <- gsea.render(AGS=ags.list, FGS=fgs.list, Lowercase = 1,
ags.gene.col = 2, ags.group.col = 3, fgs.gene.col = 2, fgs.group.col = 3,
echo=1, Ntotal = 20000, Parallelize=1)
hist(g1$estimate, breaks=100)
hist(g1$n, breaks=100)
hist(g1$p, breaks=100)
hist(g1$q, breaks=100)
```

`import.gs` *Read in an AGS or FGS test file*

Description

Imports a TAB-delimited AGS or FGS file. Checks the format of the given FGS file. Reads in unique gene/protein IDs from the given file `tbl` so that the created object can be directly submitted to `nea.render` and `gsea.render`. The function `import.gs` will be automatically called from `nea.render` if parameters `AGS` and/or `FGS` of the latter correspond to text files rather than lists.

Usage

```
import.gs(tbl, Lowercase = 1, col.gene = 2, col.set = 3, gs.type = "",
  header = FALSE)
```

Arguments

`tbl`	Input TAB-delimited FGS file.
`Lowercase`	render gene/protein IDs lower-case.
`col.gene`	Number of the column with gene/protein IDs in the input file.
`col.set`	Number of the column with set (sample, pathway etc.) IDs in the input file. Special options: `col.set = 0`: all gene/protein IDs in the file become members of a single FGS; `col.set < 0` each single gene/protein IDs becomes a separate FGS (ID of which equals the gene/protein ID).
`gs.type`	GS type, 'a' produces AGS, 'f' produces FGS.
`header`	assign logical value (TRUE/FALSE) to consider the first line as header from the input file (default:FALSE).

Value

A list with entry names that correspond AGS/FGS IDs in the input file and gene/protein IDs as elements of each entry.

See Also

`mutations2ags`, samples2ags

Examples

```
data(can.sig.go)
fpath <- can.sig.go
fgs.list <- import.gs(fpath)
summary(fgs.list)
```

`import.net` *Import a network text file*

Description

This function reads in a TAB-delimited file with the global network data needed for NEA. Checks the format of the given NETWORK file, processes unique genes from the given network file, and counts the network links.

Usage

```
import.net(tbl, Lowercase = 1, col.1 = 1, col.2 = 2, col.score = 3,
  min.score = NA, n.top = NA, echo = 1)
```

Arguments

`tbl`	Input network file.
`Lowercase`	Render gene/protein IDs lower-case.
`col.1`	Number of the column in the input file that corresponds to node 1 (gene/protein ID).
`col.2`	Number of the column in the input file that corresponds to node 2 (gene/protein ID).
`col.score`	Number of column where edge confidence score is found.
`min.score`	Minimum confidence score for an edge to be included in the network.
`n.top`	Number of edges to be included in the network, top when ranked by confidence score (is alternative to min.score).
`echo`	if execution progress should be reported.

Value

`$Ntotal`, the total number of edges in the network and `$links`, an object with members (named by gene/protein IDs) that correspond to each node in the global network. Each entry of the list `$links` contains a vector of neighbours of the respective node.

See Also

`mutations2ags`, `samples2ags`

Examples

```
data(net.kegg)
net <- import.net(net.kegg)
summary(net$links)
print(paste(names(net$links)[100], net$links[[100]], sep=": "))
```

`mutations2ags` *Create AGS from a mutation matrix*

Description

Imports a TAB-delimited file with mutations. This function creates a new list of AGSs from a table listing point (or otherwise qualitatively defined) mutations. Such a matrix M typically has size Ngenes x Nsamples, so that the current function returns a list of `length=ncol(M)`. For each of the Nsamples, AGSs are created as simple lists of all mutated genes G in a given sample S, i.e. any value X in the matrix M that satisfies condition `!is.na(X)` would be treated as a mutation. Eventual mutation types / categories are ignored. Wild type gene states in the original TAB-delimited file should be represented with NAs.

Usage

```
mutations2ags(MUT, col.mask = NA, namesFromColumn = NA, permute = FALSE)
```

Arguments

`MUT`	Matrix of size *Ngenes x Nsamples* (the both Ns are positive integers, depending on the screen scale).
`col.mask`	To include only columns with IDs that contain the specified mask. This parameter is aware of regular expression syntax, i.e. uses `grep(..., fixed = FALSE)`.
`namesFromColumn`	Number of the column (if any) that contains the gene/protein names. Note that it is only necessary if the latter are NOT the unique rownames of the matrix. This could be sometimes useful for processing redundant gene profiles with one-to-many mapping etc. Otherwise (i.e. the default), rownames shall contain gene IDs.
`permute`	If the list of AGSs should be created via random permutation of sample labels. This might be needed for testing the null hypothesis that mutated genes are randomly combined into individual genomes, while having the same frequency distribution as in the actual cohort. Since reproducing the original distribution of AGS sizes is a non-trivial set theoretical problem, the procedure is accompanied by plotting gene set sizes in actual vs. permuted AGS (the latter is usually smaller, which might be unavoidable without a sophisticated algortihm...).

Examples

```
data("tcga.gbm",package="NEArender")
dim(tcga.gbm)
ags.list <- mutations2ags(tcga.gbm, col.mask="[-.]01$")
length(ags.list)
length(unique(unlist(ags.list)))
```

nea.render *Network Enrichment Analysis (NEA)*

Description

Given the altered gene sets (AGS), functional gene sets (FGS) and the network, calculates enrichment statistics based on the number of edges (network links) that connect individual genes in each AGS-FGS pair (edges confined within AGS only or FGS only are not taken into account). Returns relevant statistics in matrices of size `length(FGS)` x `length(AGS)` (see "Value"). Each of the first three parameters can be submitted as either a text file or as a list which has been preloaded with `import.gs` and `import.net`. The latter scenario could save much time in a batch mode(see Details).

Usage

```
nea.render(AGS, FGS, NET, Lowercase = 1, ags.gene.col = 2,
  ags.group.col = 3, fgs.gene.col = 2, fgs.group.col = 3,
  net.gene1.col = 1, net.gene2.col = 2, echo = 1, graph = FALSE,
  na.replace = 0, members = FALSE, digitalize = TRUE, Parallelize = 1)
```

Arguments

AGS	Either a text file or a list of members of each AGS (see Details). Group IDs should be found in `ags.group.col` and gene IDs should be found in `ags.gene.col`. As a special option, an AGS list can also be pre-created from raw data matrices of the format "genes X samples". Such matrices can contain information on gene copy number, gene/protein expression, gene methylation etc (`samples2ags`) or point mutation data (`mutations2ags`). These two functions return AGS lists of length=ncol(Nsamples).
FGS	Either a text file or a list of members of each FGS (see Details). Group IDs should be found in `fgs.group.col` and gene IDs should be found in `fgs.gene.col`. As an alternative, this function can use single-node FGSs from the network nodes, which should be pre-created with `as_genes_fgs`. In this case each network node X becomes an FGS on its own. Respective names in the FGS list equal 'X' and the content becomes c('X'). The total length of the FGS list then equals the no. of nodes in the network. If fewer nodes are needed for such single-node analysis, then one can submit a text file in which `fgs.gene.col` and `fgs.group.col` contain identical gene IDs. This option uses `as_genes_fgs`.
NET	The global network for the analysis (see Details) .
Lowercase	render node and group IDs lower-case (default:1, i.e. 'yes').
ags.gene.col	number of the column containing AGS genes (only needed if AGS is submitted as a text file).
ags.group.col	number of the column containing group IDs (only needed if AGS is submitted as a text file).
fgs.gene.col	number of the column containing FGS genes (only needed if FGS is submitted as a text file).

fgs.group.col	number of the column containing group IDs (only needed if FGS is submitted as a text file).
net.gene1.col	number of the column containing first nodes of each network edge (only needed if NET is submitted as a text file).
net.gene2.col	number of the column containing second nodes of each network edge (only needed if NET is submitted as a text file).
echo	if messages about execution progress should appear.
graph	Plot the heat map
na.replace	replace NA values. Default=0, i.e. not to replace
members	If matrices members.fgs and members.fgs should be created in addition (time- and memory-consuming). These matrices contain lists of genes that contributed to respective AGSxFGS enrichment statistic.
digitalize	If the node ID strings should be converted to internal integer ID, and then back to present the results (can speed up the computation). Since this procedure also takes some time, setting *digitalize=TRUE* only makes sense for large computations, with a big network, many FGS and/or AGS.
Parallelize	The number of CPU cores to be used for the step "Counting actual links" (the other steps are sufficiently fast). The option is not supported on Windows.

Details

both AGS and FGS can be either
1) a list preloaded from a text file using e.g. GS=import.gs("text_file.groups"); names of the list entries are gene set IDs and the entries contain gene/protein IDs that belong to the respective set or
2) name of the file "text_file.groups" to be read from the disk using import.gs.
The TAB-delimited file "text_file.groups" should contain pre-compiled gene sets, so that gene set IDs will be found in column ags.group.col / fgs.group.col and gene IDs will be found in ags.gene.col / fgs.gene.col. Option (1) is much more efficient than (2) when nea.render has to be run multiple times. Similarly to AGS and FGS, NET could be submitted as either a list, pre-loaded with import.net, or a text TAB-delimited file where two columns net.gene1.col and net.gene2.col represent nodes of respective edges.

Value

An object, i.e. a list of elements n.actual, n.expected, chi, z, p, q, each of which is a matrix of size length(FGS) x length(AGS). The two former ones contain the number of network edges between any nodes of AGS and any nodes of FGS, respectively those observed in the actual network and expected by chance. chi are the original chi-squared network enrichment statistic values. z are respective z-scores which are normally distributed under null and are thus suitable as input to regression modelling and other parametric methods. p and q are p-values and respective FDR estimates from p.adjust(p, method="BH").

References

https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-017-1534-y

See Also

samples2ags

Examples

```
ags.list <- samples2ags(fantom5.43samples, Ntop=20, method="topnorm")
data(can.sig.go)
fpath <- can.sig.go
fgs.list <- import.gs(fpath)
summary(fgs.list)
data(net.kegg)
netpath <- net.kegg
net <- import.net(netpath)
n1 <- nea.render(AGS=ags.list[1:10], FGS=fgs.list[1:10], NET=net, graph=FALSE)
hist(n1$chi, breaks=100)
hist(n1$z, breaks=100)
hist(n1$p, breaks=100)
hist(n1$q, breaks=100)
```

---

net.kegg *A more compact alternative global network for the network enrichment analysis*

Description

A more compact alternative global network for the network enrichment analysis

Usage

```
data(net.kegg)
```

Format

A list, see import.net

Examples

```
head(net.kegg)
```

roc *ROC for NEA benchmarks*

Description

Plot ROC curve(s) for benchmarked network(s)

Usage

```
roc(tpvsfp, mode = "roc", tpr.stop = 1, fpr.stop = 1, coff.z = 1.965,
  coff.fdr = 0.1, cex.main = 1, cex.leg = 0.75, main = NA,
  use.lty = FALSE, print.stats = TRUE, sort_by_letter_and_remove = FALSE)
```

Arguments

tpvsfp	a list contaning data for the ROC curve(s) from benchmark: cross.z, cutoffs, ne, nv,tp, fp, tn, fn.
mode	either ROC curve (roc) or precision recall curve (prc). default: roc
tpr.stop	upper limit for the Y-axis (true positives). default: 1
fpr.stop	upper limit for the X-axis (false prositives), default: 1
coff.z	the point where to stop the ROC curve (in order to disable, set coff.z to below the minimal possible level)
coff.fdr	the point where to put the circle ('cross') at the ROC curve
cex.main	font size for the main title
cex.leg	font size for the legend
main	title name for the plot, default: none
use.lty	if different line types should be used for the curves, useful when the number of curves is very big (>10). default:"FALSE"
print.stats	add N(edges) and N(vertices) to the legend lines; only works if non-empty $ne and $nv elements are submitted in tpvsfp. default:"TRUE"
sort_by_letter_and_remove	enable a specific order for the curves in the legend, set by prefix letters gsub("[A-Z]\:", "", p1, fixed such as "a:network1", "b:network-2" etc., whereby a: and b: are removed from the text strings. default:"FALSE"

Details

The function roc can be called either from inside benchmark, or separately. In the latter case, it should receive the first argument as an object of the same type as benchmark can return. Generally, a ROC curve encompasses the whole interval from 0% to 100% at both axes. However in case of predictions made via network analysis, only the interval with (at least) formally significant scores is of interest (Merid et.al 2014). Therefore the benchmark results are presented as ROC curves that end at minimal acceptable confidence, set via either a z-score cut-off. Additionally, an extra point is denoted that might correspond to cross.z in roc. The scale is given in no. of tested member genes rather then as percentage of the total no. of tests. The scale ranges xlim and ylim of the plot can be reduced by submitting parameters tpr.stop and fpr.stop with values <1.

References

http://www.biomedcentral.com/1471-2105/15/308

See Also

benchmark

Examples

```
# Benchmark and plot one networks on the whole set of test GSs, using no mask:
data(can.sig.go);
fpath <- can.sig.go
gs.list <- import.gs(fpath, Lowercase = 1, col.gene = 2, col.set = 3);
data(net.kegg)
netpath <- net.kegg
net <- import.net(netpath)

b0 <- benchmark (NET = net,
 GS = gs.list,
 echo=1, graph=TRUE, na.replace = 0, mask = ".", minN = 0,
 coff.z = 1.965, coff.fdr = 0.1, Parallelize=2);
 roc(b0, coff.z = 1.64);

## Not run:
## Benchmark and plot a number of networks on GO terms and KEGG pathways separately, using masks
b1 <- NULL;
for (mask in c("kegg_", "go_")) {
b1[[mask]] <- NULL;
for (file.net in c("netpath")) {
# a series of networks can be put here: c("netpath1", "netpath2", "netpath3")
net <- import.net(netpath, col.1 = 1, col.2 = 2, Lowercase = 1, echo = 1)
b1[[mask]][[file.net]] <- benchmark (NET = net, GS = gs.list,
gs.gene.col = 2, gs.group.col = 3, net.gene1.col = 1, net.gene2.col = 2,
echo=1, graph=FALSE, na.replace = 0, mask = mask, minN = 0, Parallelize=2);
}}
par(mfrow=c(2,1));
roc(b1[["kegg_"]], coff.z = 2.57,main="kegg_");
roc(b1[["go_"]], coff.z = 2.57,main="go_");

## End(Not run)
```

---

samples2ags *Create AGS from a raw data matrix.*

Description

This function creates a number of new AGSs from a given dataset, such as gene copy number, gene/protein expression, gene methylation etc. Such matrices M typically have size Ngenes x

Nsamples, so that the current function returns a list of `length=ncol(M)`. The AGSs for each of the Nsamples are created with one of the five available methods (see parameter `method`).

Usage

```
samples2ags(m0, Ntop = NA, col.mask = NA, namesFromColumn = NA,
  method = c("significant", "top", "toppos", "topnorm", "bestp", "toprandom"),
  cutoff.q = 0.05)
```

Arguments

`m0`	input matrix.
`Ntop`	Number of top ranking genes to include into each sample-specific AGS. Mutually exclusive with "cutoff.q". A practically recommended value of Ntop could be in the range 30...300. Ntop>1000 might decrease the analysis specificity.
`col.mask`	To include only columns with IDs that contain the specified mask. Follows the regular expression syntax.
`namesFromColumn`	Number of the column (if any) that contains the gene/protein names. Note that it is only necessary of the latter are NOT the unique rownames of the matrix. This could be sometimes needed to be able to process redundant expression etc. profiles.
`method`	Method to select sample-specific genes. One of • "significant" : Using one-sided z-test, select gene values of which in the given sample *i* deviate from the mean over all the samples, requiring q-value (Benjamini-Hochberg FDR) be below *cutoff.q*.(default) • "topnorm" : similar to "significant", i.e. calculates $(x[i]-mean(x))/sd(x)$ but does not evaluate significance. Instead, top N ranked genes `Ntop` are taken into AGS. • "top" : similar to "topnorm", but $x[i] - mean(x)$ is not divided with *sd(x)*. This might help to prioritize genes with higher *mean(x)* and ignore ones with low signal. Consider also that AGSs from "top" overlap much more with each other than those from "topnorm", i.e. would be less sample-specific. • "toppos" : similar to "top", but retrives only genes with positive values of $x[i] - mean(x)$. This might be useful when the gene expression values are small counts (such as in single-cell RNA sequencing), so that considering the left part of the distribution would not bring high-quality AGS. • "toprandom" : generates lists of Ntop random genes for each AGS.
`cutoff.q`	cutoff value. Mutually exclusive with "Ntop" (default: 0.05)

Examples

```
data("fantom5.43samples", package="NEArender")
ags.list <- samples2ags(fantom5.43samples, cutoff.q = 0.01, method="significant")
```

save_gs_list *Create a TAB-delimited text file from AGS or FGS*

Description

Each line in this file represents one gene/protein from an AGS/FGS and is accompanied with respective AGS/FGS ID. This format can be used e.g. as input at web site EviNet

Usage

```
save_gs_list(gs.list, File = "gs.list.groups")
```

Arguments

gs.list	a list created with samples2ags, mutations2ags, as_genes_fgs, or import.gs.
File	output file name.

References

http://www.biomedcentral.com/1471-2105/13/226

https://www.evinet.org/

See Also

samples2ags, mutations2ags, as_genes_fgs, import.gs

Examples

```
data(net.kegg)
netpath <- net.kegg
net <- import.net(netpath);
fgs.genes <- as_genes_fgs(net);
save_gs_list(fgs.genes, File = "single_gene_ags.groups.tsv");
```

tcga.gbm *NEArender*

Description

TCGA point mutations from the glioblastoma multiforme cohort

Usage

```
data(tcga.gbm)
```

Format

A matrix with genes as rows (9660 genes with mutations reported in at least one sample) and TCGA samples as columns (291 samples). Wild type (=reference) states are denoted with NA.

Examples

```
table(is.na(tcga.gbm))
```

---

topology2nd *Higher order topology and correlation between node degrees*

Description

Auxiliary function to estimate how non-random are associations of node degrees in network edges.

Usage

```
topology2nd(NET, Lowercase = 1, col.1 = 1, col.2 = 2, echo = 1,
  main = "Higher order topology")
```

Arguments

NET	Either a text network file or an R list imported with `import.net`
Lowercase	Render gene/protein IDs lower-case.
col.1	Number of the column in the input file that corresponds to node 1 (gene/protein ID).
col.2	Number of the column in the input file that corresponds to node 2 (gene/protein ID).
echo	if execution progress should be reported.
main	The plot title.

See Also

connectivity, benchmark

Examples

```
file <-  system.file("extdata", "Connectivity.FC1_full", package = "NEArender")

 topology2nd(NET=file)
```

Index