Package ‘FGNet’

October 7, 2014

Type Package

Title Functional gene networks derived from biological enrichment analyses

Description Build and visualize functional gene networks from clustering of enrichment analyses in multiple annotation spaces. The package includes an interface to perform the analysis through David and GeneTerm Linker.

Version 2.0.0

Date 2014-03-21

Author
Sara Aibar, Celia Fontanillo, Conrad Droste and Javier De Las Rivas. Bioinformatics and Functional Genomics Group. Cancer Research Center (CiC-IBMCC, CSIC/USAL). Salamanca. Spain.

Maintainer Sara Aibar <saibar@usal.es>

Depends R (>= 2.15)

Imports igraph (>= 0.6), RCurl, hwriter, R.utils, XML

Enhances RColorBrewer, png, RDAVIDWebService

Suggests RUnit, BiocGenerics, org.Sc.sgd.db

License GPL (>= 2)

URL http://gtlinker.cnb.csic.es

LazyLoad yes

biocViews Annotation, GO, Pathways, GeneSetEnrichment, Networks,NetworkVisualization

R topics documented:

`FGNet-package` *Functional gene networks derived from biological enrichment analyses*

Description

Build and visualize functional gene/protein networks from clustering of enrichment analyses in multiple annotation spaces. The package includes an interface to perform the enrichment through David and GeneTerm Linker.

Details

Package:	FGNet
Type:	Package
License:	GPL (>= 2)

Author(s)

Author: Sara Aibar, Celia Fontanillo and Javier De Las Rivas. Bioinformatics and Functional Genomics Group. Cancer Research Center (CiC-IBMCC, CSIC/USAL). Salamanca. Spain.

If you have any issue, you can contact us at: <jrivas at usal.es>

References

[1] Fontanillo C, Nogales-Cadenas R, Pascual-Montano A, De Las Rivas J (2011) Functional Analysis beyond Enrichment: Non-Redundant Reciprocal Linkage of Genes and Biological Terms. PLoS ONE 6(9): e24289. URL: `http://gtlinker.cnb.csic.es`

[2] Huang DW, Sherman BT, Lempicki RA (2009) Bioinformatics enrichment tools: paths toward the comprehensive functional analysis of large gene lists. Nucleic Acids Res. 37(1):1-13. URL: `http://david.abcc.ncifcrf.gov/`

See Also

`report_gtLinker()` and `report_david()` are wrapper functions that include all the following steps and generate an HTML report.

The workflow to generate the functional networks include the following steps:
1. - Query/Analize (optional): Performs the analysis through GeneTerm Linker [1] or DAVID [2]. `query_gtLinker()` returns the analysis jobID and `query_david()` the url of the .txt file with the results.
2. - Get results: Retrieves the analysis results (metagroups/clusters and gene-term sets) from the server and formats them. This can be done through `getResults_gtLinker()` and `getResults_david()`
3. - Transform into incidence matrices: `toMatrix()` transforms the metagroups and gene-term sets (gtset) into genes-metagroups and genes-gtsets matrices.
The incidence matrices can also be term-based in order to build the network of associated terms. 4. - Plot the functional gene network: Network representing the common functions of the genes (or terms).
The main function is `functionalNetwork()`), but `intersectionNetwork()`
and `plotMetagroupsDistance()` are also available.

Examples

```
genesYeast <- c("ADA2", "APC1", "APC11", "APC2", "APC4", "APC5", "APC9",
"CDC16", "CDC23", "CDC26", "CDC27", "CFT1", "CFT2", "DCP1", "DOC1", "FIP1",
"GCN5", "GLC7", "HFI1", "KEM1", "LSM1", "LSM2", "LSM3", "LSM4", "LSM5",
"LSM6", "LSM7", "LSM8", "MPE1", "NGG1", "PAP1", "PAT1", "PFS2", "PTA1",
"PTI1", "REF2", "RNA14", "RPN1", "RPN10", "RPN11", "RPN13", "RPN2", "RPN3",
"RPN5", "RPN6", "RPN8", "RPT1", "RPT3", "RPT6", "SGF11", "SGF29", "SGF73",
"SPT20", "SPT3", "SPT7", "SPT8", "TRA1", "YSH1", "YTH1")

# Run analysis and generate HTML report:
# GtLinker:
org <- "Sc"
annots <- c("KEGG_Pathways")
# report_gtLinker(genesYeast, annotations=annots, organism=org, jobName="ExampleYeast")

# More examples in:
if (interactive())
    vignette("FGNet-vignette")
```

`functionalNetwork` *Create and plot the functional gene network.*

Description

Plots the gene network. Edges join nodes with common gene-term sets. Background and node color represent metagroup/clusters. White nodes are in several metagroups/clusters.

Usage

```
functionalNetwork(incidMatrices, plotType="static", returnGraph=FALSE,
plotTitle="Functional Network", vSize=12, vLabelCex=3/4,
vLayout=NULL, bgTransparency=0.4, legendMg=NULL, keepColors=TRUE,
eColor="#323232", weighted=FALSE)
```

Arguments

`incidMatrices`	list or matrix. Raw output (list) from `toMatrix`. List with slots: "gtSetsMatrix", "filteredOut" and either "metagroupsMatrix" or "clustersMatrix". If only a matrix is provided, it will be asumed to be the metagroups matrix, and all nodes will be connected to every other node in the metagroup.
`plotType`	character "static", "dynamic" or "none". "static" will generate a standard plot. "dynamic" will produce an interactive tkplot. In the interactive plot metagroups background cannot be drawn, instead the intersection network will also be shown. "none" will not plot the network.
`returnGraph`	logical. If TRUE, the igraph object containing the network is returned.
`plotTitle`	character. Title to show on the plot.
`vSize`	numeric. Vertex size.
`vLabelCex`	numeric. A numerical value giving the amount by which plotting text and symbols should be magnified relative to the default label size.
`vLayout`	2 x n matrix. Where n is the number of nodes in the graph, each column gives the (x, y)-coordinates for the corresponding node.
`bgTransparency`	numeric. Value between 0 and 1 for the transparency of the metagroups background.
`legendMg`	character. Label to show next to the metagroup/cluster id in the legend. If FALSE, legend is not shown.
`keepColors`	logical. If TRUE, it will keep the same colors for all the plots, independently of the filtered groups. Only available if metagroupGenesMatrix is the raw result from `toMatrix`.
`eColor`	character. Color for the edges.
`weighted`	logical. If TRUE, edges width will be based on the number of shared gene-term sets.

Value

Plots the functional network.
If plotType="dynamic" it also plots the intersectionNetwork.
If returnGraph=TRUE: Returns the igraph object with the network.

See Also

To see the terms included in each metagroup or cluster: `getTerms()` Full description of the package: `FGNet`

Examples

```
jobID <- 3907019
results <- getResults_gtLinker(jobID, jobName="gtLinkerExample")
incidMat <- toMatrix(results, threshold=0,
attribute=results$metagroups[,"Silhouette Width", drop=FALSE])

functionalNetwork(incidMat)
functionalNetwork(incidMat, plotType="dynamic")
getTerms(results)

# To modify the layout and plot as static network (with metagroup background)...
library(igraph)
saveLayout <- tkplot.getcoords(1)   # tkp.id (ID of the tkplot window)
functionalNetwork(incidMat, vLayout=saveLayout)

# Only return the network, without plotting
fNw <- functionalNetwork(incidMat, plotType="none", returnGraph=TRUE)
class(fNw)
betweenness(fNw)
igraph.to.graphNEL(fNw)


# Term-based network
incidMatTerms <- toMatrix(results, key="Terms")
functionalNetwork(incidMatTerms, plotType="dynamic", weighted=TRUE, eColor="grey")

# Including generic terms filterd by GtLinker from final metagroups:
incidMatTerms2 <- toMatrix(results, key="Terms", removeFiltered=FALSE)
functionalNetwork(incidMatTerms2, weighted=TRUE)

### DAVID:
genesMetabolism <- c("YGR175C", "YHR007C", "YMR202W", "YJL167W",
 "YNL280C", "YGR060W", "YGL001C", "YLR100W", "YLR056W", "YGL012W",
 "YMR015C", "YML008C", "YHR072W", "YHR190W", "YKL004W", "YBR036C",
 "YDR294C", "YDR072C", "YKL008C", "YHL003C", "YMR296C", "YDR062W",
 "YJL134W", "YOR171C", "YLR260W", "YMR298W", "YMR272C", "YPL057C",
 "YDR297W", "YBR265W", "YPL087W", "YBR183W", "YKR053C")
txtFile <- query_david(genesMetabolism)
results <- getResults_david(txtFile)

# Term-based network
incidMatTerms <- toMatrix(results, key="Terms")
functionalNetwork(incidMatTerms$clustersMatrix, plotType="dynamic")
```

getResults_david *Get results from an analysis with David.*

Description

Retrieves the clustering results from David. The .txt file URL can be obtained from query_david or from a analysis performed directly at David (http://david.abcc.ncifcrf.gov/summary.jsp) .

Usage

```
getResults_david(inputFileLocation, path = getwd(),
 jobName = "", geneLabels=NULL)
```

Arguments

`inputFileLocation`	character. URL of the file to download, or local file if already downloaded.
`path`	character. Directory in which to save the files.
`jobName`	character. Folder name and prefix for the files.
`geneLabels`	named character vector. Gene name or label to show in the plots. The vector name should contain the label to show in the plot and the content the ID used in the query.

Details

query_david() uses DAVID's API to perform the query, therefore the maximum number of genes is limited to 400 or less depending on the used ID and final URL length. In order to perform analyses with more genes, the web interface can be used (`http://david.abcc.ncifcrf.gov/summary.jsp`). Once the 'functional annotation and clustering' is done though the website, to continue the workflow through R just provide as 'inputFileLocation' the URL of the .txt downloadable file.

Value

List:
`clusters` data.frame containing the clusters and their information:

- Cluster: Cluster ID.
- EnrichmentScore: Score for the cluster.
- nGenes: Number of genes in the cluster.
- Genes: Genes in the cluster.
- Terms: Terms in the cluster.

`geneTermSets` data.frame containing the gene-term sets that support each cluster.

- Cluster: Number (id) of the cluster the gene-term set belongs to.
- Term: Term in the gene-term set.

- Category: Type of annotation of the term (i.e. GO, Kegg...)
- Genes: Genes in the gene-term set.
- Other stats: Count, PValue, List.Total, Pop.Hits, Pop.Total, Fold.Enrichment, Bonferroni, Benjamini, FDR.

`fileName` and location of the raw .txt downloaded from David.

See Also

Next step in the workflow: `toMatrix()`

Previous step in the workflow: `query_david()`

Equivalent function for GeneTerm Linker: `getResults_gtLinker`

Full description of the package: `FGNet`

Examples

```
genesYeast <- c("YBL084C", "YDL008W", "YDR118W", "YDR301W", "YDR448W",
"YFR036W", "YGL240W", "YHR166C", "YKL022C", "YLR102C", "YLR115W", "YLR127C",
"YNL172W", "YOL149W", "YOR249C")

txtFile <- query_david(genesYeast)

results <- getResults_david(txtFile)

# If the analysis results are already in a local txt file:
# getResults_david("../DavidClustering.txt")

# To add a gene label/symbol to the plots...
library(org.Sc.sgd.db)
geneLabels <- unlist(as.list(org.Sc.sgdGENENAME)[genesYeast])
names(genesYeast) <- geneLabels
results <- getResults_david(txtFile, geneLabels=genesYeast)

# To continue the workflow... (see help for further details)
incidMat <- toMatrix(results)
functionalNetwork(incidMat)
```

`getResults_gtLinker` *Get results from GeneTerm Linker.*

Description

Retrieves the metagroups and gene-term sets for the given jobID from GeneTerm Linker. The jobID can be obtained from query_gtLinker or from an analysis performed directly at GeneTerm Linker web (http://gtlinker.cnb.csic.es) .

Usage

```
getResults_gtLinker(jobID, path = getwd(), jobName = "",
alreadyDownloaded = FALSE,  keepTrying=FALSE,
serverWeb = "http://gtlinker.cnb.csic.es",
serverWS =  "http://gtlinker.cnb.csic.es:8182")
```

Arguments

`jobID`	numeric. ID of the job/analysis in GeneTerm Linker.
`path`	character. Directory in which to save the files.
`jobName`	character. Folder name and prefix for the files.
`alreadyDownloaded`	logical. If the files have already been downloaded, these will be read. Make sure to use the appropiate jobID and job name.
`keepTrying`	logical. If true, if the job has not finished, it will keep trying to get the results every few seconds.
`serverWeb`	character. GeneTerm Linker web server. It should match the web service or web address in which the analysis was performed. Available mirrors: "http://gtlinker.cnb.csic.es" and "http://cicblade.dep.usal.es:8000"
`serverWS`	character. GeneTerm Linker webservice server. Available mirrors: "http://gtlinker.cnb.csic.es:8182" and "http://cicblade.dep.usal.es:8182". If you change the webserice server, make sure to use the matching 'serverWeb'.

Value

List:
`globalMetagroups` data.frame containing the metagroups and their information:

- Size: Number of gene-term sets supporting the metagroup.
- Diameter: Maximum Cosine distance within the GeneTerm-sets of each metagroup (ranges from 0 to 1).
- Similarity: 1 - average Cosine distance within the GeneTerm-sets of each metagroup (ranges from 0 to 1). Distance and similarity calculations are done based on the genes present in the metagroups.
- Silhouette Width: Measures the compactness and proximity of multiple groups (ranges from 1 to -1). Metagroups with negative Silhouette Width usually include diverse annotations and genes with low functional coherence.
- Genes: Genes in the metagroup.
- nGenes: Number of genes in the metagroup.
- nref_list: Number of annotated genes in the reference list.
- pValue: Adjusted p-value.
- Terms: Non-generic terms in the metagroup.

`geneTermSets` data.frame containing the gene-term sets that support each metagroup.

- Metagroup: Id of the metagroup the gene-term set belongs to.
- Genes: Genes in the gene-term set.
- nGenes: Number of annotated genes in the input list. In brackets: Total number of genes in the input list.
- nref_list: Number of annotated genes in the reference list. In brackets: Total number of genes in the reference list.
- pValue: Adjusted p-value.
- Terms: Terms in the gene-term set.

See Also

Next step in the workflow: `toMatrix()`

Previous step in the workflow: `query_gtLinker()`

Equivalent function for DAVID: `getResults_david`

Full description of the package: `FGNet`

Examples

```
jobID <- 3907019
results <- getResults_gtLinker(jobID)

# To continue the workflow... (see help for further details)
incidMat <- toMatrix(results)
functionalNetwork(incidMat)
```

`getTerms` *Get terms in the metagroups/clusters.*

Description

Gets the terms in each metagroup/cluster (simplifyes the raw output from GeneTermLinker or DAVID).

Usage

```
getTerms(results)
```

Arguments

`results`	Output returned by `getResults_gtLinker()` or `getResults_david()`.

Value

`List of matrices`
: Each matrix contais the terms in each metagroup. This matrix contains only the term description. To get the term ID, annotation type, number of genes, or any other information, see the raw results returned by getResults.

See Also

Full description of the package: `FGNet`

Examples

```
## Not run:
# GeneTermLinker
jobID <- 9396814
results <- getResults_gtLinker(jobID)
getTerms(results)

## End(Not run)
```

`intersectionNetwork` *Metagroup/Cluster intersection network.*

Description

Plots a simplified version of the functional network, only with the genes in more than one metagroup/cluster.

Usage

```
intersectionNetwork(metagroupsMatrix, plotType = "dynamic",
vLayout = "kk", returnGraph = FALSE, vSize = 12, vLabelCex = 2/3,
legendMg=NULL, grPrefix="", plotTitle = "Nodes in several metagroups",
keepColors=TRUE, plotAllMg=FALSE)
```

Arguments

`metagroupsMatrix`	list or matrix. raw output (list) from `toMatrix`, or 'metagroupsMatrix' or 'clustersMatrix'.
`plotType`	character "static", "dynamic" or "none". "static" will generate a standard plot. "dynamic" will produce an interactive tkplot (metagroups background cannot be drawn). "none" will not plot the network.
`vLayout`	character. "kk" (Kamada Kawai), "circle", or "sugiyama" (hierarquical).
`returnGraph`	logical. If TRUE, the igraph object containing the network is returned.
`vSize`	numeric. Vertex size.

`vLabelCex`	numeric. A numerical value giving the amount by which plotting text and symbols should be magnified relative to the default label size.
`legendMg`	character. Label to show next to the metagroup/cluster id in the node label.
`grPrefix`	character. Prefix for the metagroup/cluster.
`plotTitle`	character. Title to show on the plot.
`keepColors`	logical. If TRUE, it will keep the same colors for all the plots, independently of the filtered groups. Only available if metagroupGenesMatrix is the raw result from `toMatrix`.
`plotAllMg`	logical. If TRUE, metagroup/cluster nodes will be plotted even if they are not connected to any more nodes.

Value

Plots the network. If returnGraph=TRUE: Returns the igraph object with the network.

See Also

Full description of the package: `FGNet`

Examples

```
jobID <- 3907019
results <- getResults_gtLinker(jobID, jobName="gtLinkerExample")
incidMat <- toMatrix(results,
 attribute=results$metagroups[,"Silhouette Width", drop=FALSE], threshold=0)

intersectionNetwork(incidMat)
intNw <- intersectionNetwork(incidMat,
 vLayout="sugiyama", plotType="static", returnGraph=TRUE)


# Term-based network
incidMatTerms <- toMatrix(results, key="Terms")
intersectionNetwork(incidMatTerms)

# Including generic terms filterd by GtLinker from final metagroups:
incidMatTerms2 <- toMatrix(results, key="Terms", removeFiltered=FALSE)
intersectionNetwork(incidMatTerms2)
```

plotMetagroupsDistance

Plots distances between metagroups.

Description

Plots the distances between metagroups taking into account the number of common genes.

Usage

```
plotMetagroupsDistance(incidenceMatices)
```

Arguments

`incidenceMatices`	Object returned by `toMatrix()`.

Value

Plot and distance matrix.

See Also

Full description of the package: `FGNet`

Examples

```
results <- getResults_gtLinker(jobID=1963186, jobName="gtLinker_ej")
incidMat <- toMatrix(results,
 attribute=results$metagroups[,"Silhouette Width", drop=FALSE])
plotMetagroupsDistance(incidMat)
```

`query_david` *Query DAVID.*

Description

Sends a new analysis to DAVID [1] to perform the functional enrichment and clustering.

Usage

```
query_david(genes, geneIdType = "ENSEMBL_GENE_ID",
 annotations = c("GOTERM_BP_ALL", "GOTERM_MF_ALL", "GOTERM_CC_ALL",
 "KEGG_PATHWAY", "INTERPRO"), email=NULL,
 argsWS = c(overlap=4L, initialSeed=4L, finalSeed=4L, linkage=0.5, kappa=35L))
```

Arguments

`genes`	character vector. List of genes to analyze.
`geneIdType`	character vector. Type of gene identifier. Web API: ENSEMBL_GENE_ID, ENTREZ_GENE_ID, GENE_SYMBOL, UNIPROT_ID... For more, check DAVID's API documentation. Web Service: run `getIdTypes(DAVIDWebService$new(email=...))`
`annotations`	character vector. Annotation spaces for the functional analysis. Web API: check DAVID's API documentation. Web Service: run `getAllAnnotationCategoryNames(DAVIDWebService$new(email=...))`.

`email`	character. If provided, the query will be performed though DAVID's Web Service (recommended). Requires registration.
`argsWS`	named integer vector. Additional arguments for the clustering. Only available using the web service.

Details

If an email is provided, the query will be performed through the web service, if not through the API:

Web service: Recommended option. Requiers registration at `http://david.abcc.ncifcrf.gov/webservice/register.htm`.

API: Allows to perform a small query without registering.
Maximum number of genes: 400. It can be less depending on the ID types.
The default arguments for the clustering performed through the API are: `web_API_defaults <- c(overlap=3L, initialSe`
More details and full list of gene ID types and annotations are available at: `http://david.abcc.ncifcrf.gov/content.jsp?file=DAVID_API.html`.

Website: If the *functional annotation and clustering* has been performed directly at DAVID's website (`http://david.abcc.ncifcrf.gov/summary.jsp`) this function is not required. Provide the file (or the URL of the file) containing the results of the analysis to `getResults_david()`.

Value

Returns the url/location of the text file containing the results of the analysis.

References

[1] Huang DW, Sherman BT, Lempicki RA (2009) Bioinformatics enrichment tools: paths toward the comprehensive functional analysis of large gene lists. Nucleic Acids Res. 37(1):1-13.

See Also

Next step in the workflow: `getResults_david()`

Equivalent function for GeneTerm Linker: `query_gtLinker()`

Full description of the package: `FGNet`

Examples

```
## Not run:
genesYeast <- c("YBL084C", "YDL008W", "YDR118W", "YDR301W", "YDR448W",
"YFR036W", "YGL240W", "YHR166C", "YKL022C", "YLR102C", "YLR115W", "YLR127C",
"YNL172W", "YOL149W", "YOR249C")

# Default:
txtFile <- query_david(genesYeast)

# Web service (requires email):
```

```
txtFile_ws <- query_david(genesYeast, email="example@email.com")

# The API has different default arguments than the web service.
# To obtain exactly the same results using the web service, use:
API_defaults <- c(overlap=3L, initialSeed=3L, finalSeed=3L, linkage=0.5, kappa=50L)
query_david(genesYeast, email="...", argsWS=API_defaults)


# To continue the workflow... (see help for further details)
results <- getResults_david(txtFile)
incidMat <- toMatrix(results)
functionalNetwork(incidMat)

## End(Not run)
```

---

query_gtLinker *Query GeneTerm Linker*

Description

Sends a new analysis to GeneTerm Linker [1] to perform its functional enrichment.

Usage

```
query_gtLinker(genes, organism = "Hs",
annotations = c("GO_Biological_Process", "GO_Molecular_Function",
"GO_Cellular_Component", "KEGG_Pathways", "InterPro_Motifs"),
minSupport = 4, serverWS = "http://gtlinker.cnb.csic.es:8182")
```

Arguments

`genes`	character vector. List of genes to analyze.
`organism`	character. "Hs" (Homo sapiens) or "Sc" (Saccharomyces cerevisiae).
`annotations`	character vector. Annotation spaces for the functional analysis. Available spaces: "GO_Biological_Process", "GO_Molecular_Function", "GO_Cellular_Component", "KEGG_Pathways", "InterPro_Motifs".
`minSupport`	numeric. Minimum number of genes per group.
`serverWS`	character. GeneTerm Linker webservice server. Available mirrors: "http://gtlinker.cnb.csic.es:8182" and "http://cicblade.dep.usal.es:8182". If you change the webserice server, make sure to use the matching 'serverWeb' in the following step.

Value

Returns jobID, the ID of the analysis, which will be required for the next steps.

References

[1] Fontanillo C, Nogales-Cadenas R, Pascual-Montano A, De Las Rivas J (2011) Functional Analysis beyond Enrichment: Non-Redundant Reciprocal Linkage of Genes and Biological Terms. PLoS ONE 6(9): e24289.

See Also

Next step in the workflow: `getResults_gtLinker()`

Equivalent function for DAVID: `query_david()`

Full description of the package: `FGNet`

Examples

```
## Not run:
genesYeast <- c("ADA2", "APC1", "APC11", "APC2", "APC4", "APC5", "APC9",
"CDC16", "CDC23", "CDC26", "CDC27", "CFT1", "CFT2", "DCP1", "DOC1", "FIP1",
"GCN5", "GLC7", "HFI1", "KEM1", "LSM1", "LSM2", "LSM3", "LSM4", "LSM5",
"LSM6", "LSM7", "LSM8", "MPE1", "NGG1", "PAP1", "PAT1", "PFS2", "PTA1",
"PTI1", "REF2", "RNA14", "RPN1", "RPN10", "RPN11", "RPN13", "RPN2", "RPN3",
"RPN5", "RPN6", "RPN8", "RPT1", "RPT3", "RPT6", "SGF11", "SGF29", "SGF73",
"SPT20", "SPT3", "SPT7", "SPT8", "TRA1", "YSH1", "YTH1")

# jobID <- query_gtLinker(genesYeast, organism = "Sc")

jobID <- query_gtLinker(genesYeast, organism = "Sc",
 annotations = c("KEGG_Pathways"), minSupport = 3)


# To continue the workflow... (see help for further details)
jobID <- 3907019
results <- getResults_gtLinker(jobID)
incidMat <- toMatrix(results)
functionalNetwork(incidMat)

## End(Not run)
```

`report_gtLinker` *Generate the Functional Network report for a gene list performing the functional analysis through GeneTerm Linker or DAVID.*

Description

Perform the functional analysis and clustering and generate an HTML report containing the functional network.

Usage

```
report_gtLinker(genes=NULL, organism="Hs",
annotations=c("GO_Biological_Process","GO_Molecular_Function",
"GO_Cellular_Component", "KEGG_Pathways", "InterPro_Motifs"),
minSupport=4, jobID=NULL, alreadyDownloaded=FALSE,
downloadGOtree=TRUE, path=getwd(), jobName=NULL, threshold=0,
serverWeb="http://gtlinker.cnb.csic.es",
serverWS="http://gtlinker.cnb.csic.es:8182")

report_david(genes=NULL, geneIdType="ENSEMBL_GENE_ID",
annotations=c("GOTERM_BP_ALL", "GOTERM_MF_ALL", "GOTERM_CC_ALL",
"KEGG_PATHWAY", "INTERPRO"), email=NULL,
argsWS = c(overlap=4L, initialSeed=4L, finalSeed=4L,
linkage=0.5, kappa=35L), inputFileLocation=NULL, path=getwd(),
jobName=NULL, threshold=0, geneLabels=NULL, downloadGOtree=TRUE)
```

Arguments

	Common to both tools:
`genes`	character vector. List of genes to analyze.
`annotations`	character vector. Annotation spaces used for the functional analysis. Available spaces for GtLinker: "GO_Biological_Process","GO_Molecular_Function", "GO_Cellular_Component", "KEGG_Pathways", "InterPro_Motifs". Examples for DAVID: "GOTERM_BP_ALL", "GOTERM_MF_ALL", "GOTERM_CC_ALL", "KEGG_PATHWAY", "INTERPRO"... For more, see `query_david()`.
`path`	character. Directory in which to save the files.
`jobName`	character. Folder name and file prefix for the files.
`threshold`	numeric. Threshold to filter the metagroups/clusters. Those with a value under the threshold will not be plotted. By default, GeneTerm Linker metagroups are filtered based on their *Silhouette Width*, DAVID's clusters based on their *Enrichment Score*.
`downloadGOtree`	logical. if TRUE download the go term trees png. If FALSE the report will contain the link to the web tool (faster).
	Specific for GeneTerm Linker:
`organism`	character. "Hs" (Homo sapiens) or "Sc" (Saccharomyces cerevisiae).
`minSupport`	numeric. Minimum number of genes per group.
`jobID`	numeric. ID of the job/analysis in GeneTerm Linker (from `query_gtLinker` or the ID from a query in the web).
`alreadyDownloaded`	logical. If the files have already been downloaded, these will be read. Make sure to use the appropiate jobID and job name.
`serverWS`	character. GeneTerm Linker webservice server. Available mirrors: "http://gtlinker.cnb.csic.es:8182" and "http://cicblade.dep.usal.es:8182". 'serverWS' and 'serverWeb' should match.

`serverWeb`	character. GeneTerm Linker webservice server. Available mirrors: "http://gtlinker.cnb.csic.es" and "http://cicblade.dep.usal.es:8000"
	Specific for DAVID:
`geneIdType`	character vector. Type of gene id for the genes provided. Sample ID types: GENE_SYMBOL, ENSEMBL_GENE_ID, ENTREZ_GENE_ID, UNIPROT_ID. For more, see `query_david()`.
`inputFileLocation`	character. URL of the file to download, or local file if already downloaded.
`email`	character. If provided, the query will be performed though DAVID's Web Service (recommended). Requires registration at `http://david.abcc.ncifcrf.gov/webservice/register.htm`.
`argsWS`	named integer vector. Additional arguments for the clustering. Only available using the web service.
`geneLabels`	named character vector. Gene name or label to show in the plots. The vector name should contain the label to show in the plot and the content the ID used in the query.

Details

The report functions are wrappers that includes the following steps:
1. - Query (optional): Performs the analysis through GeneTerm Linker or David. `query_gtLinker()` returns the analysis jobID and `query_david()` the *.txt file* with the raw David results.
2. - Get results: Retrieve the metagroups/clusters and gene-term sets from the server. This is done through the functions `getResults_gtLinker()` and `getResults_gtLinker()`
3. - Transform into incidence matrices (`toMatrix()`): Transforms the metagroups and gene-term sets (gtset) into genes-metagroups and genes-gtset incidence matrices.
4. - Generate the HTML report, which includes the functional network (`functionalNetwork()`).

See Also

Full description of the package: `FGNet`

Examples

```
## Not run:
##### Runs analysis and generates HTML report:

# GtLinker:
genesYeast <- c("ADA2", "APC1", "APC11", "APC2", "APC4", "APC5", "APC9",
"CDC16", "CDC23", "CDC26", "CDC27", "CFT1", "CFT2", "DCP1", "DOC1", "FIP1",
"GCN5", "GLC7", "HFI1", "KEM1", "LSM1", "LSM2", "LSM3", "LSM4", "LSM5",
"LSM6", "LSM7", "LSM8", "MPE1", "NGG1", "PAP1", "PAT1", "PFS2", "PTA1",
"PTI1", "REF2", "RNA14", "RPN1", "RPN10", "RPN11", "RPN13", "RPN2", "RPN3",
"RPN5", "RPN6", "RPN8", "RPT1", "RPT3", "RPT6", "SGF11", "SGF29", "SGF73",
"SPT20", "SPT3", "SPT7", "SPT8", "TRA1", "YSH1", "YTH1")
organism <- "Sc"
annotations <- c("KEGG_Pathways")
```

```
# report_gtLinker(genesYeast, annotations=annotations,
  organism=organism, jobName="ExampleYeast")

# or, for an already executed query:
report_gtLinker(jobID="3907019", jobName="ExampleYeast", downloadGOtree=FALSE)

# David:
genesYeast <- c("YBL084C", "YDL008W", "YDR118W", "YDR301W", "YDR448W",
"YFR036W", "YGL240W", "YHR166C", "YKL022C", "YLR102C", "YLR115W", "YLR127C",
"YNL172W", "YOL149W", "YOR249C")
report_david(genesYeast, jobName="ExampleYeast_David") #add: email="your@email.com"

# Add gene labels (i.e. Gene symbol to show instead of ID)
library(org.Sc.sgd.db)
geneLabels <- unlist(as.list(org.Sc.sgdGENENAME)[genesYeast])
names(genesYeast) <- geneLabels
report_david(genesYeast, geneLabels=genesYeast,
 annotations = c("GOTERM_BP_ALL", "GOTERM_MF_ALL", "KEGG_PATHWAY", "INTERPRO"),
 downloadGOtree=FALSE)


## End(Not run)
```

`toMatrix` *Transforms into group-genes incidence matrices.*

Description

Transforms the raw results from GeneTermLinker or DAVID into incidence matrices.

Usage

```
toMatrix(results, attribute=NULL, threshold=0,
key="Genes", removeFiltered=NULL)
```

Arguments

`results`	list or data.frame. *results* returned by `getResults_gtLinker` or `getResults_david`.
`attribute`	data.frame. Attribute (column from results) to filter the metagroups/clusters.
`threshold`	numeric. Metagroups/cluster with *attribute* lower than this threshold will be filtered.
`key`	"Genes" or "Terms". To build genes- or terms-based networks.
`removeFiltered`	logical. If FALSE, it includes generic terms filtered by GeneTerm Linker from final metagroups. Only available when building a terms network with GeneTerm Linker.

Value

List:

`metagroupsMatrix or clustersMatrix`	Incidende matrix: Genes or Terms in each metagroup or cluster.
`gtSetsMatrix`	Incidende matrix: Genes or Terms in each gene-term set
`filteredOut`	Metagroups/clusters which where filtered out and therefore not included in the incidence matrices. NULL if none.

See Also

Next step in the workflow: `functionalNetwork()` (Plots)

Previous step in the workflow: `getResults_gtLinker()` or `getResults_david()`

Full description of the package: `FGNet`

Examples

```
jobID <- 3907019
results <- getResults_gtLinker(jobID)
incidMat <- toMatrix(results)

# Filtering (threshold)
incidMat <- toMatrix(results,
 attribute=results$metagroups[,"Silhouette Width", drop=FALSE], threshold=0.2)

incidMat$filteredOut
head(incidMat$metagroupsMatrix)
head(incidMat$gtSetsMatrix)

functionalNetwork(incidMat)

# Filtering (keyword)
keywords <- c("rna")
selectedGroups <- sapply(getTerms(results),
function(x)
any(grep(paste("(", paste(keywords, collapse="|") ,")",sep=""), tolower(x))))

resultsCbind <- results
resultsCbind$metagroups <- cbind(results$metagroups,
 selectedKeywords=as.numeric(selectedGroups))

matSelectedGroups <- toMatrix(resultsCbind$geneTermSets,
 attribute=resultsCbind$metagroups[,"selectedKeywords", drop=FALSE], threshold=1)

functionalNetwork(matSelectedGroups)

# Term-based network
incidMatTerms <- toMatrix(results, key="Terms")
```

```
functionalNetwork(incidMatTerms, plotType="dynamic")

# Including generic terms filterd by GtLinker from final metagroups:
incidMatTerms2 <- toMatrix(results, key="Terms", removeFiltered=FALSE)
intersectionNetwork(incidMatTerms2)
```

Index