Package 'FGNet'

November 6, 2026

Type Package

Title Functional Gene Networks derived from biological enrichment analyses

Description Build and visualize functional gene and term networks from clustering of enrichment analyses in multiple annotation spaces. The package includes a graphical user interface (GUI) and functions to perform the functional enrichment analysis through DAVID, GeneTerm Linker, gage (GSEA) and topGO.

Version 3.47.0

Date 2022-04-08

Author Sara Aibar, Celia Fontanillo, Conrad Droste and Javier De Las Rivas.

Maintainer Sara Aibar <saibar@usal.es>

Depends R (>= 4.2.0)

VignetteBuilder knitr

Imports igraph (>= 0.6), hwriter, R.utils, XML, plotrix, reshape2, RColorBrewer, png, methods, stats, utils, graphics, grDevices

Suggests RCurl, gage, topGO, GO.db, reactome.db, RUnit, BiocGenerics, org.Sc.sgd.db, knitr, rmarkdown, AnnotationDbi, BiocManager

License GPL (>= 2)

URL http://www.cicancer.org

LazyLoad yes

biocViews Annotation, GO, Pathways, GeneSetEnrichment, Network, Visualization, FunctionalGenomics, NetworkEnrichment, Clustering

git_url https://git.bioconductor.org/packages/FGNet

git_branch devel

git_last_commit b133ae0

git_last_commit_date 2026-04-28

Repository Bioconductor 3.24

Date/Publication 2026-11-05

Contents

FGNet-package . . . 2
analyzeNetwork . . . 4
clustersDistance . . . 5
fea2incidMat . . . 6
fea_gage . . . 8
fea_gtLinker . . . 10
fea_topGO . . . 13
FGNet_GUI . . . 15
FGNet_report . . . 16
format_david . . . 18
format_results . . . 20
functionalNetwork . . . 21
getTerms . . . 24
keywordsTerm . . . 25
organisms . . . 25
plotGoAncestors . . . 26
readGeneTermSets . . . 27

Index **28**

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

Description

Build and visualize functional gene and term networks from clustering of enrichment analyses in multiple annotation spaces. The package includes a graphical user interface (GUI) and functions to perform the functional enrichment analysis through DAVID, GeneTerm Linker, gage (GSEA) and topGO.

Details

Package:	FGNet
Type:	Package
Version:	3.0
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/

[3] Alexa A, and Rahnenfuhrer J (2010) topGO: Enrichment analysis for Gene Ontology. R package version 2.16.0. URL: http://www.bioconductor.org/packages/release/bioc/html/topGO.html

[4] Luo W, Friedman MS, Shedden K, Hankenson KD, Woolf PJ (2009) GAGE: generally applicable gene set enrichment for pathway analysis. BMC Bioinformatics. 10:161. URL: http://www.bioconductor.org/packages/release/bioc/html/gage.html

See Also

`FGNet_GUI()` provides a Graphical User Interface (GUI) to most of the functionalities of the package: Performing a Functional Enrichment Analysis (FEA) of a list of genes, and analyzing it through the functional networks.

1. The Functional Enrichment Analysis can be performed through several tools:

   - GeneTerm Linker [2]: `fea_gtLinker()` & `fea_gtLinker_getResults()` (Requires internet connection)
   - topGO [3]: `fea_topGO()` (Only supports GO. For offline use requires having installed the required database packages)
   - GAGE [4]: `fea_gage()` (GSEA analysis. For offline use requires gene sets or installed database packages)

     There are also a few functions to import the results from a previous FEA analysis: `format_david()`, `format_results()` and `readGeneTermSets()`.

2. `FGNet_report()`: automatically generates a report with the default network options. It includes the following steps, wich can be executed individually to personalize or explore the networks:

   1. `fea2incidMat()`: Transforms the FEA output into incidence matrices. These function determines wether the network will be gene- or term-based.
   2. `functionalNetwork()`: Generates and plots the functional networks. These networks can be further explored by `analyzeNetwork()` and `clustersDistance()`.

      Other auxiliary functions: `getTerms()`, `keywordsTerm()`, `plotGoAncestors()`

      For more info see the package tutorial: `vignette("FGNet-vignette")`

Examples

```
## Not run:
# GUI:
FGNet_GUI()


# 1. FEA:
geneList <- c("YBL084C", "YDL008W", "YDR118W", "YDR301W", "YDR448W", "YFR036W",
    "YGL240W", "YHR166C", "YKL022C", "YLR102C", "YLR115W", "YLR127C", "YNL172W",
    "YOL149W", "YOR249C")
```

```
library(org.Sc.sgd.db)
geneLabels <- unlist(as.list(org.Sc.sgdGENENAME)[geneList])

# Optional: Gene expression
geneExpr <- setNames(c(rep(1,10),rep(-1,5)), geneLabels)

# Choose FEA tool...
# results <- fea_david(geneList, geneLabels=geneLabels, email="example@email.com")
results <- fea_gtLinker_getResults(jobID=3907019)

# 2 A) Report:
FGNet_report(results, geneExpr=geneExpr)

# 2 B) Step by step:
# 2.1. Create incidence matrices:
incidMat <- fea2incidMat(results)
incidMat_terms <- fea2incidMat(results, key="Terms")

# 2.2. Explore networks:
functionalNetwork(incidMat, geneExpr=geneExpr)
functionalNetwork(incidMat_terms, plotType="bipartite", plotOutput="dynamic")
getTerms(results)

nwStats <- analyzeNetwork(incidMat)
clustersDistance(incidMat)

## End(Not run)
```

`analyzeNetwork` *Analyze Functional Network*

Description

Analyzes the degree and betweenness of the genes in the functional network.

Usage

```
analyzeNetwork(incidMatrices, fNw = NULL, plotOutput = TRUE, colors = NULL)
```

Arguments

`incidMatrices`	list or matrix. Output from `fea2incidMat` or the equivalent incidence matrices.
`fNw`	list. Return from `functionalNetwork` to avoid recalculating.
`plotOutput`	logical. Wether to plot the degree and betweenness boxplots.
`colors`	vector. Colors for the metagroups

Value

List:

- degree, betweenness: Degree and Betweenness of the nodes in the global network (commonClusters) and within each cluster/metagroup (subsets of commonGtSets network). The degree is given as percentage, normalized based on the total number of nodes of the network. i.e. a value of 90 in a network of 10 nodes, would mean the actual degree of the node is 9: it is conneded to 9 nodes (90% of 10)).
- transitivity: Transitivity of the networks.
- betweennessMatrix: Betweenness of each node in each cluster.
- hubsList: Nodes selected as potential hubs in the global network and within each cluster/metagroup (nodes with betweenness over 75% in the given network/subnetwork).
- intraHubsCount: Number of times each node was selected as potential intra-cluster hub.

See Also

Overview of the package: FGNet

Package tutorial: `vignette("FGNet-vignette")`

Examples

```
## Not run:
# Previous Steps
jobID <- 3907019
results <- fea_gtLinker_getResults(jobID)
incidMat <- fea2incidMat(results, filterAttribute="Silhouette Width")


# Plot node degree and betweensess
analyzeNetwork(incidMat)

# Get stats without plotting
nwStats <- analyzeNetwork(incidMat, plotOutput=FALSE)
names(nwStats)
nwStats$hubsTable

## End(Not run)
```

---

clustersDistance *Plots distances between metagroups.*

Description

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

Usage

```
clustersDistance(incidenceMatices, mgCols = NULL, clustMethod="average")
```

Arguments

incidenceMatices	Object returned by `fea2incidMat()`.
mgCols	Colors for the metagroups.
clustMethod	Clustering method. Character string (i.e. "single", "complete", "average") for function `hclust()` (argument 'method').

Value

Plot and distance matrix.

See Also

Full description of the package: FGNet

Examples

```
## Not run:
results <- fea_gtLinker_getResults(jobID=1963186, jobName="gtLinker_example")
incidMatrices <- fea2incidMat(results)
clustersDistance(incidMatrices)

## End(Not run)
```

`fea2incidMat` *Transforms FEA output into incidence matrices.*

Description

Transforms the Funtional Enrichment Analysis (FEA) results into cluster-gene incidence matrices.

Usage

```
fea2incidMat(feaResults, key = "Genes", sepChar = NULL, clusterColumn = NULL,
    filterAttribute = NULL, filterOperator = "<", filterThreshold = 0,
    removeFilteredGtl = NULL)
```

Arguments

`feaResults`	list or data.frame/matrix. Output from one of the FEA functions.
`key`	"Genes" or "Terms". To build gene- or term-based networks.
`sepChar`	character. Character separating genes or terms in the same field. By default: "," for genes and ";" for terms.
`clusterColumn`	character. Name of the column that contains the value to group gene-term sets. Only required if it is different than "Cluster" or "Metagroup".
`filterAttribute`	character or data.frame. Attribute to filter the clusters/metagroups. Filtered clusters/metagroups will not be included in the matrices (and subsequent networks). Its value should be the data.frame column to use for filtering. It can be provided as character (column name) or data.frame (subset of the data.frame with drop=FALSE).
`filterOperator`	character. Logical operator used for filtering. i.e. ">" (bigger than), "<=" (smaller or equal than), "==" (equal), "!=" (different), "%%" (included in),... The evaluation order is left to right: filterAttribute ">" filterThreshold, will filter out clusters with filter attribute bigger than the threshold.
`filterThreshold`	numeric. Sets the value to compare to.
`removeFilteredGtl`	logical. Only used by GeneTerm Linker term network. If FALSE, it includes generic terms filtered by GeneTerm Linker from final metagroups.

Value

List:

`clustersMatrix` or `metagroupsMatrix`
: Incidende matrix with the genes or Terms in each cluster or metagroup.

`gtSetsMatrix`
: Incidende matrix with the genes or Terms in each gene-term set

`filteredOut`
: Clusters or metagroups which where filtered out and therefore not included in the incidence matrices. NULL if none.

See Also

Next step in the workflow: `functionalNetwork()`

Overview of the package: `FGNet`

Package tutorial: `vignette("FGNet-vignette")`

Examples

```
## Not run:
jobID <- 3907019
results <- fea_gtLinker_getResults(jobID)
incidMat <- fea2incidMat(results)

# Filtering (threshold)
incidMat <- fea2incidMat(results,
    filterAttribute="Silhouette Width", filterThreshold=0.2)

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

functionalNetwork(incidMat)

# Term-based network
incidMatTerms <- fea2incidMat(results, key="Terms")
functionalNetwork(incidMatTerms, plotOutput="dynamic")

# Including generic terms filterd by GtLinker from final metagroups:
incidMatTerms <- fea2incidMat(results, key="Terms",removeFilteredGtl=FALSE)
functionalNetwork(incidMatTerms, plotOutput="dynamic", plotType="bipartite")

# Filtering by 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 <- fea2incidMat(resultsCbind,
    filterAttribute="selectedKeywords", filterThreshold=1)

functionalNetwork(matSelectedGroups)
getTerms(results)[selectedGroups]
```

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

`fea_gage` *FEA - GAGE*

Description

Performs the functional enrichment analysis and clustering through GAGE [1] (GSEA).

Usage

```
fea_gage(eset, refSamples, compSamples, geneIdType, geneLabels=NULL,
    organism = "Hs",
    annotations = c("GO_BP", "GO_MF", "GO_CC", "REACTOME"),
    geneSets = NULL,
    sameDirection = FALSE,
    onlyEssentialTerms = TRUE,
    compareType = "as.group",
    jobName = NULL, ...)
```

Arguments

`eset`	expressionSet or expression matrix.
`refSamples`	numeric. Index of the samples to use as reference (control).
`compSamples`	numeric. Index of the samples to analyze.
`geneIdType`	character. Type of gene identifier should be the same as the one provided in the geneSets, or available in the organism package.
`geneLabels`	named character vector. Gene name or label to use in the report/plots instead of the original gene ID. The vector names should be the gene ID and the content of the vector the gene label. The resulting geneTermSets table will contain the original gene ID column (geneIDs) and the label column (Genes).
`organism`	two letter code for the organism. See: `data(organisms);organisms`
`annotations`	character vector. Annotation spaces to select from the provided geneSets. Set to NULL to use the geneSets as is (i.e. geneSets not split/named by annotation)
`geneSets`	geneSets. If NULL geneSets are calculated automatically based on the organism, gene ID and annotations. The geneSets can also be provided from a previous execution or loaded from a .gtm file. i.e.: `readList("c2.cp.v4.0.symbols.gmt")`
`sameDirection`	logical. Should all the genes in the geneSet be altered in the same direction (up/down)?
`onlyEssentialTerms`	logical. Wether to simplify the results and keep only the essential terms in the clusters.
`compareType`	character: 'as.group', 'unpaired', '1ongroup'... See GAGE for details.
`jobName`	character. Folder name and prefix for the files.
`...`	other arguments to pass to GAGE.

Value

Invisible list with the folowing fields:
`queryArgs` list with the arguments for the query.

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

- Cluster: Cluster ID.
- nGenes: Number of genes in the cluster.
- dir: Direction in which the term/pathway is altered (Up/Down).
- 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.
- essentialSet: Logical. Is the pathway selected as essential?
- dir: Direction in which the term/pathway is altered (Up/Down).
- Terms: Term in the gene-term set.
- Genes: Genes in the gene-term set.
- GenesIDs: In case GeneLabels was provided, original gene ID.
- Other stats provided by GAGE: p.geomean, stat.mean, p.val, q.val, set.size

`fileName`: .txt file with the FEA results. `genesFC`: Fold change.

References

[1] Luo W, Friedman MS, Shedden K, Hankenson KD, Woolf PJ (2009) GAGE: generally applicable gene set enrichment for pathway analysis. BMC Bioinformatics. 10:161. URL: http://www.bioconductor.org/packages/release/bioc/html/gage.html

See Also

- `fea_gtLinker()` & `fea_gtLinker_getResults()` (Requires internet connection)
- `fea_topGO()`
  To import results from a previous/external FEA analysis: `format_david()`, `format_results()` and `readGeneTermSets()`.

Overview of the package: `FGNet`

Package tutorial: `vignette("FGNet-vignette")`

Examples

```
## Not run:
 # Load expressionSet:
library(gage)
data(gse16873)

# Load gene labels?
library(org.Hs.eg.db)
geneSymbols <- select(org.Hs.eg.db,columns="SYMBOL",keytype="ENTREZID",
    keys=rownames(gse16873))
head(geneSymbols)
```

```
table(table(geneSymbols$ENTREZID)) # All need to be unique identifiers

geneLabels <- geneSymbols$SYMBOL
names(geneLabels) <- geneSymbols$ENTREZID
head(geneLabels)

# FEA:
results <- fea_gage(eset=gse16873,
                    refSamples=grep('HN',colnames(gse16873), ignore.case =T),
                    compSamples=grep('DCIS',colnames(gse16873), ignore.case=T),
                    geneIdType="ENTREZID", geneLabels=geneLabels, organism="Hs",
                    annotations="REACTOME")


# To continue the workflow... (see help for further details)
FGNet_report(results)

incidMat <- fea2incidMat(results)
functionalNetwork(incidMat)

## End(Not run)
```

`fea_gtLinker` *FEA - Gene-Term Linker*

Description

Performs the functional enrichment analysis and clustering through Gene-Term Linker [1] (requires internet connection).

Since Gene-Term Linker takes a while to analyze the gene list, the process has been splitted in two steps:

1. fea_gtLinker(): Submits the query
2. fea_gtLinker_getResults(): Retrieves the results of the analysis. It might take a few minutes for the results to become available.

Usage

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

fea_gtLinker_getResults(jobID = NULL, organism = NULL, jobName = NULL,
    alreadyDownloaded = FALSE, keepTrying = FALSE,
    serverWeb = "http://gtlinker.cnb.csic.es",
    serverWS = "http://gtlinker.cnb.csic.es:8182")
```

Arguments

fea_gtLinker():

`geneList`	character vector. List of genes to analyze.

`annotations` character vector. Annotation spaces for the functional analysis.
Available values: "GO_Biological_Process", "GO_Molecular_Function", "GO_Cellular_Component", "InterPro_Motifs".

`minSupport` numeric. Minimum number of genes per group.

common arguments:

`serverWS` character. GeneTerm Linker webservice server.
Available mirrors: "http://gtlinker.cnb.csic.es:8182" If you change the webserice server, make sure to use the matching 'serverWeb' in the following step.

`organism` character. "Hs" (Homo sapiens) or "Sc" (Saccharomyces cerevisiae).

fea_gtLinker_getResults():

`jobID` numeric. ID of the job/analysis in GeneTerm Linker.

`jobName` character. Folder name and prefix for the files.

`alreadyDownloaded` logical. If the files have already been downloaded, these will be read instead of downloaded again.

`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"

Value

fea_gtLinker() returns the jobID of the analysis

fea_gtLinker_getResults() returns an invisible list with the folowing fields:

`queryArgs` list with the arguments for the query.

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

- Metagroup: Metagroup ID.
- 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.

fileName .txt file with the formatted FEA results.

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

See Also

Other FEA tools:

- fea_gage()
- fea_topGO()

  To import results from a previous/external FEA analysis: format_david(), format_results() and readGeneTermSets().

Overview of the package: FGNet

Package tutorial: vignette("FGNet-vignette")

Examples

```
## Not run:
### Execute FEA:
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")
# Optional expression (1=UP, -1=DW):
genesYeastExpr <- setNames(c(rep(1,29), rep(-1,30)), genesYeast)

# Submit query
jobID <- fea_gtLinker(geneList=genesYeast,organism="Sc")
jobID


### Get results from FEA:
jobID <- 3907019 # job ID of the query
results <- fea_gtLinker_getResults(jobID=jobID)
```

```
# To continue the workflow... (see help for further details))
incidMat <- fea2incidMat(results)
functionalNetwork(incidMat)

# Or full report
FGNet_report(results, geneExpr=genesYeastExpr)

## End(Not run)
```

---

fea_topGO *FEA - topGO*

Description

Performs the functional enrichment analysis through topGO [1].

Usage

```
fea_topGO(geneList, geneIdType = "ENSEMBL", geneLabels=NULL, organism = "Hs",
    annotations = c("GO_BP", "GO_MF", "GO_CC"), evidence=NULL,
    genesUniverse = NULL, refPackage = NULL,
    geneID2GO = NULL, nodeSize = 5, pValThr = 0.01, testStat = NULL,
    jobName = NULL)
```

Arguments

geneList	character vector. List of genes to analyze.
geneIdType	character. Type of gene identifier should be available for the organism package.
geneLabels	named character vector. Gene name or label to use in the report/plots instead of the original gene ID. The vector names should be the gene ID and the content of the vector the gene label. The resulting geneTermSets table will contain the original gene ID column (geneIDs) and the label column (Genes).
organism	two letter code for the organism. See: `data(organisms);organisms`
annotations	character vector. Annotation spaces for the functional analysis. Accepted values: "GO_BP", "GO_MF", "GO_CC".
evidence	character vector. Required evidence code for GO annotations. If NULL no filtering is done (all annotations are used). For full list, see the organism "EVIDENCE" keys: i.e `keys(org.Hs.eg.db, keytype="EVIDENCE")`. For non-comprehensive code description: `data(GOEvidenceCodes)`.
genesUniverse	character vector. List of genes used for background (i.e. all genes available in the chip).
refPackage	character. Name of the package to use for calculating the genes universe. A Chip package is recommended. If NULL the genes universe is set as all the genes available in the organism package.
geneID2GO	GO gene sets. If NULL it is calculated automatically.
nodeSize	numeric. Minimum size of GO terms. TopGo authors recommend 5-10 for more stable results, 1 for no prune.
pValThr	numeric. P-value threshold.
testStat	classicCount from toGO. If NULL: GOFisherTest is used.
jobName	character. Folder name and prefix for the files.

Value

Invisible list with the folowing fields:
`queryArgs` list with the arguments for the query.

`clusters` Empty list. only for compatibility.

`geneTermSets` data.frame containing the gene-term sets.

- Ont: Ontology to wich the term belongs (BP, MF or CC)
- Terms: Term in the gene-term set.
- Genes: Genes in the gene-term set.
- GenesIDs: In case GeneLabels was provided, original gene ID.
- Other stats provided by topGO: Annotated, Significant, Expected, classic.

`fileName` .txt file with the formatted FEA results.

References

[1] Adrian Alexa and Jorg Rahnenfuhrer (2010) topGO: Enrichment analysis for Gene Ontology. R package version 2.16.0. URL: http://www.bioconductor.org/packages/release/bioc/html/topGO.html

See Also

Other FEA tools:

- `fea_gtLinker()` & `fea_gtLinker_getResults()` (Requires internet connection)
- `fea_gage()`
  To import results from a previous/external FEA analysis: `format_david()`, `format_results()` and `readGeneTermSets()`.

Overview of the package: `FGNet`

Package tutorial: `vignette("FGNet-vignette")`

Examples

```
## Not run:

# Load/format gene list:
geneList <- c("YBL084C", "YDL008W", "YDR118W", "YDR301W", "YDR448W", "YFR036W",
    "YGL240W", "YHR166C", "YKL022C", "YLR102C", "YLR115W", "YLR127C", "YNL172W",
    "YOL149W", "YOR249C")

library(org.Sc.sgd.db)
geneLabels <- unlist(as.list(org.Sc.sgdGENENAME)[geneList])

geneExpr <- setNames(c(rep(1,10),rep(-1,5)), geneLabels)

# FEA (using directly the gene names):
results <- fea_topGO(geneLabels, geneIdType="GENENAME", organism="Sc")

# FEA (using the gene ID, and replacing it by the label after the FEA):
results <- fea_topGO(geneList, geneIdType="ENSEMBL",
    geneLabels=geneLabels, organism="Sc")
```

```
# To continue the workflow... (see help for further details)
FGNet_report(results, geneExpr=geneExpr)

incidMat <- fea2incidMat(results, geneExpr=geneExpr)
functionalNetwork(incidMat)



## End(Not run)
```

FGNet_GUI *FGNet graphical user interface*

Description

Provides a graphical user interface (GUI) to most FGNet functionalities.

Usage

```
FGNet_GUI(geneList = NULL)
```

Arguments

geneList	vector. If provided, assigns the value to the genes field. It can be a character vector containing the gene list, or a named numeric vector with the gene expression.

Details

To generate the functional network, first execute or import the Functional Analisis of the gene list with one of the tools in Tab "1 - FEA", then generate the network or the report in Tab "2 - Network".

Value

Opens the GUI. No value is returned. The results of the analyses will be saved in the current working directory.

Note

Available for Windows and Linux. The current version of the GUI is not available for Mac OS X Snow Leopard.

See Also

Overview of the package: FGNet

Package tutorial: `vignette("FGNet-vignette")`

Examples

```
## Not run:


FGNet_GUI()

# To directly input a gene list (i.e. from a previous analysis):
geneList <- c("YBL084C", "YDL008W", "YDR118W", "YDR301W", "YDR448W", "YFR036W",
    "YGL240W", "YHR166C", "YKL022C", "YLR102C", "YLR115W", "YLR127C", "YNL172W",
    "YOL149W", "YOR249C")
# Optional gene expression
geneExpr <- setNames(c(rep(1,10),rep(-1,5)), geneList)

FGNet_GUI(geneExpr)


## End(Not run)
```

FGNet_report *FGNet report*

Description

Generates an HTML report with several views of the Functional Network and complementary analyses.

Usage

```
FGNet_report(feaResults, geneExpr = NULL, plotExpression = "border",
onlyGoLeaves = TRUE, plotGoTree = TRUE,
filterAttribute = NULL, filterOperator = NULL, filterThreshold = NULL)
```

Arguments

feaResults	list or data.frame/matrix. Output from one of the FEA functions.
geneExpr	numeric. Named vector with the relative expression value of the gene (node). 0 is taken as reference, positive values will be plotted red, negative values green.
plotExpression	character. Determines the way to plot the expression: "border" adds a red or green border to the node, "fill" colors the whole with the expression color instead of the metagroup color.
onlyGoLeaves	logical. If TRUE only terminal GO terms (leaves in the ontology tree) will be included in the cluster list.
plotGoTree	logical. If TRUE plots containing the terms in their position within the GO ontology (tree) will be generated.
filterAttribute	character or data.frame. Attribute to filter the clusters/metagroups. Filtered clusters/metagroups will not be included in the matrices (and subsequent networks). Its value should be the data.frame column to use for filtering. It can be provided as character (column name) or data.frame (subset of the data.frame with drop=FALSE).

`filterOperator` character. Logical operator used for filtering. i.e. ">" (bigger than), "<=" (smaller or equal than), "==" (equal), "!=" (different), "%%" (included in),... The evaluation order is left to right: filterAttribute ">" filterThreshold, will filter out clusters with filter attribute bigger than the threshold.

`filterThreshold`
numeric. Sets the value to compare to.

Value

Generates the HTML report in the current directory.

Warning

Reactome ID change depending on the database version. Links to reactome website are created to ease the analysis, but in case the version used for the enrichment and the website's do not match, broken links or misleading links might appear.

See Also

Functional enrichment analysis functions:

- `fea_gtLinker()` & `fea_gtLinker_getResults()` (Requires internet connection)
- `fea_gage()`
- `fea_topGO()`
  To import results from a previous/external FEA analysis: `format_david()`, `format_results()` and `readGeneTermSets()`.

Overview of the package: `FGNet`

Package tutorial: `vignette("FGNet-vignette")`

Examples

```
## Not run:

# Report with diferent tools:

##########################
# DAVID & TopGO
geneList <- c("YBL084C", "YDL008W", "YDR118W", "YDR301W", "YDR448W", "YFR036W",
    "YGL240W", "YHR166C", "YKL022C", "YLR102C", "YLR115W", "YLR127C", "YNL172W",
    "YOL149W", "YOR249C")
geneExpr <- setNames(c(rep(1,10),rep(-1,5)), geneList)

library(org.Sc.sgd.db)
geneLabels <- unlist(as.list(org.Sc.sgdGENENAME)[geneList])
names(geneExpr) <- geneLabels[names(geneExpr)]

# DAVID
results_David <- fea_david(geneList, geneLabels=geneLabels, email="example@email.com")
FGNet_report(results_David, geneExpr=geneExpr)

# TopGO
results_topGO <- fea_topGO(geneList, geneIdType="ENSEMBL",
    geneLabels=geneLabels, organism="Sc")
FGNet_report(results_topGO, geneExpr=geneExpr)
```

```
##########################
# Gage
library(gage); data(gse16873)
results_gage <- fea_gage(eset=gse16873,
    refSamples=grep('HN',colnames(gse16873), ignore.case =T),
    compSamples=grep('DCIS',colnames(gse16873), ignore.case=T),
    geneIdType="ENTREZID", organism="Hs", annotations="REACTOME")
FGNet_report(results_gage)

##########################
# Gene-Term Linker:

# Execute new query:
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")
# Optional expression (1=UP, -1=DW):
genesYeastExpr <- setNames(c(rep(1,29), rep(-1,30)), genesYeast)


jobID <- fea_gtLinker(geneList=genesYeast,organism="Sc")

# Load existing query:
jobID <- 3907019

results_gtLinker <- fea_gtLinker_getResults(jobID=jobID)
FGNet_report(results_gtLinker, geneExpr=genesYeastExpr)



## End(Not run)
```

`format_david` *Format DAVID output*

Description

Format DAVID 'functional annotation and clustering' output to use with FGNet.

Usage

```
format_david(fileName,
jobName = NULL,
geneLabels = NULL,
moveFile = FALSE,
downloadFile=TRUE)
```

Arguments

`fileName`	character. URL or local file with the results of a DAVID analysis, for example, performed at DAVID's Website (http://david.abcc.ncifcrf.gov/summary.jsp). In case of local file, it should be the absolute path to the .txt file (whole location from root to the file including file name: "C:\\Documents\\23424203.txt", "/home/user/2342342.txt").
`jobName`	character. Folder name and prefix for the formatted files.
`geneLabels`	named character vector. Gene name or label to use in the report/plots instead of the original gene ID. The vector names should be the gene ID and the content of the vector the gene label. The resulting geneTermSets table will contain the original gene ID column (geneIDs) and the label column (Genes).
`moveFile`	logical. If TRUE the original file is moved to the new location. If FALSE, the file is copied.
`downloadFile`	logical. If TRUE, the result files are saved in the current directory (required to generate report).

Value

Invisible list with the folowing fields:
`queryArgs` list with the arguments for the query.

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

- Cluster: Cluster ID.
- nGenes: Number of genes in the cluster.
- ClusterEnrichmentScore: Score for the cluster.
- Genes: Genes in the cluster.
- Terms: Terms in the cluster.
- keyWordsTerm: Term is the most representative of the terms in the cluster based on keywords.

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

- Cluster: Number (id) of the cluster the gene-term set belongs to.
- ClusterEnrichmentScore: Score for the cluster. Same value for all terms in each cluster.
- Category: Type of annotation of the term (i.e. GO, Kegg...)
- Terms: Term in the gene-term set.
- Genes: Genes in the gene-term set.
- GenesIDs: In case GeneLabels was provided, original gene ID.
- Other stats: Count, PValue, List.Total, Pop.Hits, Pop.Total, Fold.Enrichment, Bonferroni, Benjamini, FDR.

`fileName` .txt file with the formatted FEA results.

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

Overview of the package: FGNet

Package tutorial: vignette("FGNet-vignette")

Examples

```
# Select file:
txtFile <- "http://david.abcc.ncifcrf.gov/data/download/901234901248.txt"
txtFile <- paste(file.path(system.file('examples', package='FGNet')),
    "DAVID_Yeast_raw.txt", sep="/")

# Read:
results <- format_david(txtFile, jobName="DavidAnalysis")


# To continue the workflow... (see help for further details)
getTerms(results)
incidMat <- fea2incidMat(results)
functionalNetwork(incidMat)

?FGNet_report
```

---

format_results *Format FEA results from external tools.*

Description

Format the functional analysis results from external tools to use with FGNet.

Usage

```
format_results(fileName, newFileName = NULL, clusterCol = NULL,
    geneCol = NULL, geneSep = NULL, termDescCol = NULL, termIDCol = NULL,
    termCatCol = NULL, termCat = NULL, termSep = NULL,
    tool = "Imported text file", simplifyGage = TRUE, ...)
```

Arguments

fileName	character. File name with the FEA results.
newFileName	character. Name for the formatted files.
clusterCol	character. Name of the column to use for clustering.
geneCol	character. Name of the column with the genes.
geneSep	character. Character separating diferent genes in the same field (i.e. ",", ";", ...)
termDescCol	character. Name of the column with the terms description.
termIDCol	character. Name of the column with the terms ID.
termCatCol	character. Name of the column with the terms type/category.
termCat	character. Name of the annotation type if it is common to all gene-term sets. Provide either termCatCol or termCat, not both.

termSep	character. Character separating diferent terms in the same field (i.e. ",", ";", ...)
tool	character. Tool used for the FEA (row name from `data(FEA_tools); FEA_tools`)
simplifyGage	logical. For internal use, only for GAGE. Determines wether to keep non essential terms in the final clusters.
...	Further argumets to pass to "read.table"

Value

Saves the formatted file and returns an invisible list with the appropiate format to use with FGNet_report() and fea2incidMat() (fields "clusters", "geneTermSets" and "fileName").

See Also

Overview of the package: FGNet

Package tutorial: `vignette("FGNet-vignette")`

Examples

```
## Not run:

results <- format_results("/home/user/feaResults.txt", clusterCol="Cluster",
    geneCol="Genes", termDescCol="Terms", sep="\t")


## End(Not run)
```

---

functionalNetwork *Creates and plots the functional gene network.*

Description

Plots the functional networks.

The default network links genes to genes, or terms to terms. The bipartite network links genes or terms to their clusters.

Usage

```
functionalNetwork(incidMatrices, plotType = c("default", "bipartite")[1],
    plotOutput = "static", plotTitle = "Functional Network",
    plotTitleSub = NULL, legendPrefix = NULL, legendText = NULL,
    geneExpr = NULL, plotExpression = c("border", "fill"),
    vExprColors=c(neg="#008000", zero="white", pos="#FF2020"),
    vSize = 12, vLabelCex = 2/3, vLayout = NULL, keepColors = TRUE,
    bgTransparency = 0.4, eColor = "#323232", eWidth=NULL, weighted = FALSE,
    keepAllNodes = FALSE, plotAllMg = FALSE)
```

Arguments

incidMatrices	list or matrix. Raw output (list) from fea2incidMat: a list with slots: "gtSetsMatrix", "filteredOut" and either "metagroupsMatrix" or "clustersMatrix". If only a matrix is provided, it will be asumed to be the clusters matrix, and all nodes will be connected to every other node in the metagroup.
plotType	"default" or "bipartite". Default network: Nodes are either genes or terms. Edges join nodes in common gene-term sets. Background and node color represent cluster/metagroup. White nodes are in several clusters/metagroups. Bipartite network: Nodes are genes or terms (circles) and their clusters (squares). By default it keeps only the genes or terms in more than one cluster or metagroup, which represents a simplified version of the functional network. Node shape is only available in the "static" output.
plotOutput	"static", "dynamic" or "none". "static" will generate a standard R plot. "dynamic" will produce an interactive tkplot (metagroups background cannot be drawn). "none" will not plot the network.
plotTitle	character. Title to show on the plot.
plotTitleSub	character. Text to show at the bottom of the plot (sub-title).
legendPrefix	character. Label to show next to the cluster/metagroup id in the legend. In the bipartite network the legens replaces the cluster node label.
legendText	character. Description of each cluster (shown as the legend). If FALSE, legend is not shown.
geneExpr	numeric. Named vector with the relative expression value of the gene (node). 0 is taken as reference, positive values will be plotted red, negative values green.
plotExpression	character. Determines the way to plot the expression: "border" adds a red or green border to the node, "fill" colors the whole with the expression color instead of the metagroup color.
vExprColors	character. Vector with the colors for expression: first color for negative values, second for zero, and third for positive.
vSize	numeric. Vertex size. If named, it allows to set a value for each gene. Name as "default" to set a default value, otherwise the default value is the mean.
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 or character. Where n is the number of nodes in the graph, each column gives the (x, y)-coordinates for the corresponding node. The bipartite network accepts "kk" (Kamada Kawai), "circle", or "sugiyama" (hierarquical).
keepColors	logical. If TRUE, it will keep the same colors for all the plots, independently of the filtered groups. Only available if incidMatrices is the raw result from fea2incidMat().
bgTransparency	numeric. Value between 0 and 1 for the transparency of the metagroups background (only default network).
eColor	character. Color for the edges.
eWidth	numeric. Edge width. Not to plot edges, set eWidth=0 or eColor=NA.
weighted	logical. If TRUE, edges width will be based on the number of shared gene-term sets.
keepAllNodes	logical. Only used in bipartite network. If FALSE, nodes in only one cluster are not plotted. If TRUE, all nodes in the clusters are shown.

plotAllMg — logical. Only used in bipartite network. If FALSE, non-connected clusters are not plotted. If TRUE, all non-filtered clusters are shown.

Value

Plots the functional networks.

An invisible list with the igraph networks and incidence matrices, to collect it assign it to a variable.

See Also

Previous step in the workflow: `fea2incidMat()`

To see the terms included in each cluster or metagroup: `getTerms()`

Overview of the package: `FGNet`

Package tutorial: `vignette("FGNet-vignette")`

Examples

```
#####################################################
# Previous steps
# Set gene list:
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")
# Optional gene expression
genesYeastExpr <- setNames(c(rep(1,29), rep(-1,30)), genesYeast) # 1=UP, -1=DW

## Not run:
# FEA:
# jobID <- query_gtLinker(genesYeast, organism = "Sc")
jobID <- 3907019
results <- fea_gtLinker_getResults(jobID)

#####################################################
# Gene-based networks:
incidMat <- fea2incidMat(results, filterAttribute="Silhouette Width")

functionalNetwork(incidMat, geneExpr=genesYeastExpr)
functionalNetwork(incidMat, plotType="bipartite",
plotOutput="dynamic", vSize=c(default=10, GLC7=20, PTA1=20))

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, plotType="bipartite")

# Only return the network, without plotting
fNw <- functionalNetwork(incidMat, plotOutput="none")
class(fNw)
names(fNw)
```

```
betweenness(fNw$iGraph$commonClusters)

################################################
# Term-based network
incidMat_terms <- fea2incidMat(results, key="Terms")
functionalNetwork(incidMat_terms, weighted=TRUE, plotOutput="dynamic")
functionalNetwork(incidMat_terms, plotType="bipartite", plotOutput="dynamic",
    plotAllMg=TRUE)
functionalNetwork(incidMat_terms, plotType="bipartite", plotOutput="dynamic",
    keepAllNodes=TRUE)

# Including generic terms filterd by GtLinker from final metagroups:
incidMat_terms2 <- fea2incidMat(results, key="Terms", removeFiltered=FALSE)
functionalNetwork(incidMat_terms2, weighted=TRUE)

## End(Not run)
```

`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(feaResults, returnValue = "description")
```

Arguments

`feaResults`	Output returned by any of the fea functions.
`returnValue`	"description" Returns term description, e.g. "GO" returns GO term IDs.

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:
results <- fea_gtLinker_getResults(jobID=1963186)
getTerms(results)

## End(Not run)
```

keywordsTerm *Select keyword term*

Description

Selects a term as the most representative of the terms in the cluster based on keywords.

Usage

```
keywordsTerm(termsDescriptions, nChar = 30)
```

Arguments

termsDescriptions	List with the terms in each cluster. Output from getTerms.
nChar	numeric. Maximum number of chars to show in the term. If the selected term is longer, it will be trimmed.

Value

Character vector with the term selected for each cluster.

See Also

Overview of the package: FGNet

Package tutorial: `vignette("FGNet-vignette")`

Examples

```
## Not run:
# Previous Steps: FEA
results <- fea_gtLinker_getResults(3907019)

# Select keywords
keywordsTerm(getTerms(results), nChar=100)

## End(Not run)
```

organisms *FGNet data*

Description

FGNet package includes the following data files: organisms, groupTypes and FEA_tools

Usage

```
data(organisms)
organisms

data(groupTypes)
groupTypes

data(FEA_tools)
FEA_tools

data(GOEvidenceCodes)
GOEvidenceCodes
```

Value

- organisms: Matrix with the supported organisms' name and package.
- groupTypes: Matrix with the group types supported by FGNet (cluster, metagroup and gene-term set).
- FEA_tools: Matrix with info about the FEA tools supported by FGNet.
- GOEvidenceCodes: Matrix with info about GO evidence codes. (from http://geneontology.org/page/guide-go-evidence-codes, on nov. 26, 2014.)

---

plotGoAncestors *Plot GO term ancestors*

Description

Plots the ancestors in the tree ontology for the given GO terms.

Usage

```
plotGoAncestors(goIds, tColor = NULL, ontology = NULL,
    plotOutput = "static", nCharTerm = 50, nSize = NULL, labelCex = NULL,
    asp = NULL, fileName = NULL, height = 1000)
```

Arguments

goIds	character vector. GO IDs of the terms to plot.
tColor	character. Color for the term (i.e. based on expression).
ontology	character. If character determines which ontology to plot ("BP"", "MF" or "CC"").
plotOutput	"static", "dynamic" or "none". "static" will generate a standard R plot. "dynamic" will produce an interactive tkplot. "none" will not plot the network.
nCharTerm	numeric. Max term size (number of characters). Longer terms will be trimmed.
nSize	numeric. Determines the node size.
labelCex	numeric. Determines the node label size.
fileName	character. If provided, the plot is saved as png with this fileName.
asp	character. If fileName is provided, asp argument for plot.
height	numeric. If fileName is provided, height argument for png().

Value

An invisible list with the nodes identified as leaves (leaves) and the graph (iGraph).

See Also

Overview of the package: FGNet

Package tutorial: vignette("FGNet-vignette")

Examples

```
plotGoAncestors(c("GO:0000152","GO:0043234", "GO:0044446", "GO:0043227"))

# plotGoAncestors(c("GO:0051603", "GO:0019941", "GO:0051128","GO:0044265"), plotOutput="dynamic")


# From analysis:
txtFile <- paste(file.path(system.file('examples', package='FGNet')),
    "DAVID_Yeast_raw.txt", sep="/")
results <- format_david(txtFile, jobName="DavidAnalysis")

plotGoAncestors(getTerms(results, returnValue="GO")$"Cluster 7", ontology="MF")
```

---

readGeneTermSets *Read gene-term sets*

Description

Reads a file containing gene-term sets (formatted output from FEA) and transform it into clusters/metagroups and geneTermSets tables to imput to FGNet.

Usage

```
readGeneTermSets(fileName, tool = NULL, simplifyGage = TRUE)
```

Arguments

fileName	character. File name of the .txt file.
tool	character. Tool used for the FEA (row name from data(FEA_tools); FEA_tools)
simplifyGage	logical. Only for GAGE: Wether to simplify the results and keep only the essential terms in the clustes.

Value

List formated in the same way as the fea functions.

See Also

Overview of the package: FGNet

Package tutorial: vignette("FGNet-vignette")

Index

∗ **Functional analysis**
FGNet-package, 2
∗ **GO**
FGNet-package, 2
∗ **GeneSet enrichment**
FGNet-package, 2
∗ **Networks**
FGNet-package, 2
∗ **annotation**
FGNet-package, 2
∗ **dynamic**
FGNet-package, 2
∗ **graphs**
FGNet-package, 2
∗ **interface**
FGNet-package, 2
∗ **package**
FGNet-package, 2
∗ **pathways**
FGNet-package, 2
∗ **visualization**
FGNet-package, 2

adjMatrix *(*fea2incidMat*)*, 6
analyzeNetwork, 4
analyzeNetwork(), *3*

clustersDistance, 5
clustersDistance(), *3*

fea, *27*
fea functions, *24*
fea2incidMat, *4*, 6, *22*
fea2incidMat(), *3*, *5*, *21–23*
fea_gage, 8
fea_gage(), *3*, *12*, *14*, *17*
fea_gtLinker, 10
fea_gtLinker(), *3*, *9*, *14*, *17*
fea_gtLinker_getResults *(*fea_gtLinker*)*, 10
FEA_tools *(*organisms*)*, 25
fea_topGO, 13
fea_topGO(), *3*, *9*, *12*, *17*
FGNet, *5–7*, *9*, *12*, *14*, *15*, *17*, *20*, *21*, *23–25*, *27*
FGNet *(*FGNet-package*)*, 2
FGNet-package, 2
FGNet_GUI, 15
FGNet_GUI(), *3*
FGNet_report, 16
FGNet_report(), *3*, *21*
format_david, 18
format_david(), *3*, *9*, *12*, *14*, *17*
format_results, 20
functionalNetwork, *4*, 21
functionalNetwork(), *3*, *7*

GAGE, *3*, *8*, *9*
getResults_david *(*format_david*)*, 18
getResults_gtLinker *(*fea_gtLinker*)*, 10
getTerms, 24, *25*
getTerms(), *3*, *23*
GOEvidenceCodes *(*organisms*)*, 25
groupTypes *(*organisms*)*, 25

hclust(), *5*

intersectionNetwork *(*functionalNetwork*)*, 21
Invisible, *9*, *14*, *19*
invisible, *11*, *21*, *23*, *27*

keywordsTerm, 25
keywordsTerm(), *3*

organisms, 25

plotGoAncestors, 26
plotGoAncestors(), *3*
plotMetagroupsDistance *(*clustersDistance*)*, 5

query_gtLinker *(*fea_gtLinker*)*, 10

readGeneTermSets, 27
readGeneTermSets(), *3*, *9*, *12*, *14*, *17*
report_david *(*FGNet_report*)*, 16
report_gtLinker *(*FGNet_report*)*, 16

toMatrix *(*fea2incidMat*)*, 6
topGO, *3*, *14*