Package: FELLA (via r-universe)

November 25, 2026

Type Package

Title Interpretation and enrichment for metabolomics data

Version 1.34.0

Date 2019-10-19

Maintainer Sergio Picart-Armada <sergi.picart@upc.edu>

Description Enrichment of metabolomics data using KEGG entries. Given a set of affected compounds, FELLA suggests affected reactions, enzymes, modules and pathways using label propagation in a knowledge model network. The resulting subnetwork can be visualised and exported.

License GPL-3

LazyLoad yes

Imports methods, igraph, Matrix, KEGGREST, plyr, stats, graphics, utils

Depends R (>= 3.5.0)

Suggests shiny, DT, magrittr, visNetwork, knitr, BiocStyle, rmarkdown, testthat, biomaRt, org.Hs.eg.db, org.Mm.eg.db, AnnotationDbi, GOSemSim

VignetteBuilder knitr

RoxygenNote 6.1.1

Encoding UTF-8

biocViews Software, Metabolomics, GraphAndNetwork, KEGG, GO, Pathways, Network, NetworkEnrichment

Collate 'AllArguments.R' 'AllClasses.R' 'AllMethods.R' 'generateResultsTable.R' 'generateEnzymesTable.R' 'generateResultsGraph.R' 'exportResults.R' 'addGOToGraph.R' 'buildGraphFromKEGGREST.R' 'buildDataFromGraph.R' 'defineCompounds.R' 'doc-data.R' 'doc-package.R' 'runHypergeom.R' 'runDiffusion.R' 'runPagerank.R' 'enrich.R' 'get-.R' 'is-.R' 'launchApp.R' 'list-.R' 'loadKEGGdata.R' 'plotBipartite.R' 'plotGraph.R' 'plotLegend.R'

Config/pak/sysreqs libglpk-dev libpng-dev libxml2-dev libssl-dev zlib1g-dev

Repository Bioconductor 3.24

Date/Publication 2026-11-23 22:40:28 UTC

RemoteUrl https://github.com/bioc/FELLA

RemoteRef RELEASE_3_24

RemoteSha 3fb68a167b8f3ff30fc03482013c42ca3cdbc775

Contents

D.diffusion-class . . . 3
D.hypergeom-class . . . 3
D.keggdata-class . . . 3
D.pagerank-class . . . 4
data-funs . . . 4
enrich-funs . . . 7
export-funs . . . 13
FELLA . . . 18
FELLA.DATA-class . . . 19
FELLA.sample . . . 20
FELLA.USER-class . . . 20
getBackground . . . 21
getCom . . . 22
getExcluded . . . 23
getGraph . . . 24
getInfo . . . 25
getInput . . . 25
getMatrix . . . 26
getName . . . 27
getPscores . . . 27
getStatus . . . 28
getSums . . . 29
getValid . . . 29
input.sample . . . 30
is.FELLA.DATA . . . 31
is.FELLA.USER . . . 31
launchApp . . . 32
listApprox . . . 33
listCategories . . . 33
listInternalDatabases . . . 34
listMethods . . . 34
plotLegend . . . 35
U.diffusion-class . . . 35
U.hypergeom-class . . . 36
U.pagerank-class . . . 36
U.userinput-class . . . 37

Index **38**

D.diffusion-class *An internal S4 class for the diffusion data*

Description

An internal S4 class for the diffusion data

Slots

`matrix` Numeric (dense) matrix [optional]

`rowSums` Numeric named vector with rowSums internal data

`squaredRowSums` Numeric named vector with squaredRowSums internal data

D.hypergeom-class *An internal S4 class for the binary matrix (hypergeometric test)*

Description

An internal S4 class for the binary matrix (hypergeometric test)

Slots

`matrix` Binary sparse matrix

D.keggdata-class *An internal S4 class to represent the KEGG graph and related files*

Description

An internal S4 class to represent the KEGG graph and related files

Slots

`graph` KEGG graph

`id2name` Mapping list: KEGG ID to KEGG name (can contain multiple hits)

`pvalues.size` Numeric matrix for the evaluation of CC through their size

`id` List with character vectors for KEGG categories

`status` Character that specifies the current status of this S4 class

D.pagerank-class *An internal S4 class for the PageRank data*

Description

An internal S4 class for the PageRank data

Slots

`matrix` Numeric (dense) matrix [optional]

`rowSums` Numeric named vector with rowSums internal data

`squaredRowSums` Numeric named vector with squaredRowSums internal data

data-funs *Parse, build and load the KEGG knowledge model*

Description

Function `buildGraphFromKEGGREST` makes use of the KEGG REST API (requires internet connection) to build and return the curated KEGG graph.

Function `buildDataFromGraph` takes as input the KEGG graph generated by `buildGraphFromKEGGREST` and writes the KEGG knowledge model in the desired permanent directory.

Function `loadKEGGdata` loads the internal files containing the KEGG knowledge model into a `FELLA.DATA` object.

In general, `generateGraphFromKEGGREST` and `generateDataFromGraph` are one-time executions for a given organism and knowledge model, in this precise order. On the other hand, the user needs to run `loadKEGGdata` in every new R session to load such model into a `FELLA.DATA` object.

Usage

```
buildGraphFromKEGGREST(organism = "hsa", filter.path = NULL)

buildDataFromGraph(keggdata.graph = NULL, databaseDir = NULL,
    internalDir = TRUE, matrices = c("hypergeom", "diffusion",
    "pagerank"), normality = c("diffusion", "pagerank"),
    dampingFactor = 0.85, niter = 100)

loadKEGGdata(databaseDir = tail(listInternalDatabases(), 1),
    internalDir = TRUE, loadMatrix = NULL)
```

Arguments

organism	Character, KEGG code for the organism of interest
filter.path	Character vector, pathways to filter. This is a pattern matched using regexp. E.g: "01100" to filter the overview metabolic pathway in any species
keggdata.graph	An **igraph** object generated by the function buildGraphFromKEGGREST
databaseDir	Character containing the directory to save KEGG files. It is a relative directory inside the library location if internalDir = TRUE. If left to NULL, an automatic name containing the date, organism and the KEGG release is generated.
internalDir	Logical, should the directory be internal in the package directory?
matrices	A character vector, containing any of these: "hypergeom", "diffusion", "pagerank"
normality	A character vector, containing any of these: "diffusion", "pagerank"
dampingFactor	Numeric value between 0 and 1 (none inclusive), damping factor d for PageRank (page.rank)
niter	Numeric value, number of iterations to estimate the p-values for the CC size. Between 10 and 1e3.
loadMatrix	Character vector to choose if heavy matrices should be loaded. Can contain: "diffusion", "pagerank"

Details

In function buildGraphFromKEGGREST, The user specifies (i) an organism, and (ii) patterns matching pathways that should not be included as nodes. A graph object, as described in [Picart-Armada, 2017], is built from the comprehensive KEGG database [Kanehisa, 2017]. As described in the main vignette, accessible through browseVignettes("FELLA"), this graph has five levels that represent categories of KEGG nodes. From top to bottom: pathways, modules, enzymes, reactions and compounds. This knowledge representation is resemblant to the one formerly used by MetScape [Karnovsky, 2011], in which enzymes connect to genes instead of modules and pathways. The necessary KEGG annotations are retrieved through KEGGREST R package [Tenenbaum, 2013]. Connections between pathways/modules and enzymes are inferred through organism-specific genes, i.e. an edge is added if a gene connects both entries. However, in order to enrich metabolomics data, the user has to pass the graph object to buildDataFromGraph to obtain the FELLA.USER object. All the networks are handled with the igraph R package [Csardi, 2006].

Using buildDataFromGraph is the second step to use the FELLA package. The knoledge graph is used to compute other internal variables that are required to run any enrichment. The main point behind the enrichment is to provide a small part of the knowledge graph relevant to the supplied metabolites. This is accomplished through diffusion processes and random walks, followed by a statistical normalisation, as described in [Picart-Armada, 2017]. When building the internal files, the user can choose whether to store (i) matrices for each provided method, and (ii) vectors derived from such matrices to use the parametric approaches. These are optional but enable (i) faster permutations and custom metabolite backgrounds, and (ii) parametric approaches. WARNING: diffusion and PageRank matrices in (i) can allocate up to 250MB each. On the other hand, the niter parameter controls the amount of trials to approximate the distribution of the connected component size under uniform node sampling. For further info, see the option thresholdConnectedComponent in the details from ?generateResultsGraph. Regarding the destination, the user can specify the

name of the directory. Otherwise a name containing the creation date, the organism and the KEGG release will be used. The database can be stored within the library path or in a custom location.

Function `loadKEGGdata` returns a `FELLA.DATA` object from any of the databases generated by `FELLA.DATA`. This object is the starting point of any enrichment using `FELLA`. In case the user built the matrices for "diffusion" and "pagerank", he or she can choose to load them. Further detail on the methods can be found in [Picart-Armada, 2017]. The matrices allow a faster computation and the definition of a custom background, but use up to 250MB of memory each.

Value

`buildGraphFromKEGGREST` returns the curated KEGG graph (class **igraph**)

`buildDataFromGraph` returns `invisible(TRUE)` if successful. As a side effect, the directory `outdir` is created, containing the internal data.

`loadKEGGdata` returns the `FELLA.DATA` object that contains the KEGG knowledge representation.

References

Kanehisa, M., Furumichi, M., Tanabe, M., Sato, Y., & Morishima, K. (2017). KEGG: new perspectives on genomes, pathways, diseases and drugs. Nucleic acids research, 45(D1), D353-D361.

Karnovsky, A., Weymouth, T., Hull, T., Tarcea, V. G., Scardoni, G., Laudanna, C., ... & Athey, B. (2011). Metscape 2 bioinformatics tool for the analysis and visualization of metabolomics and gene expression data. Bioinformatics, 28(3), 373-380.

Tenenbaum, D. (2013). KEGGREST: Client-side REST access to KEGG. R package version, 1(1).

Chang, W., Cheng, J., Allaire, JJ., Xie, Y., & McPherson, J. (2017). shiny: Web Application Framework for R. R package version 1.0.5. https://CRAN.R-project.org/package=shiny

Picart-Armada, S., Fernandez-Albert, F., Vinaixa, M., Rodriguez, M. A., Aivio, S., Stracker, T. H., Yanes, O., & Perera-Lluna, A. (2017). Null diffusion-based enrichment for metabolomics data. PLOS ONE, 12(12), e0189012.

See Also

class `FELLA.DATA`

Examples

```
## Toy example
## In this case, the graph is not built from current KEGG.
## It is loaded from sample data in FELLA
data("FELLA.sample")
## Graph to build the database (this example is a bit hacky)
g.sample <- FELLA:::getGraph(FELLA.sample)
dir.tmp <- paste0(tempdir(), "/", paste(sample(letters), collapse = ""))
## Build internal files in a temporary directory
buildDataFromGraph(
keggdata.graph = g.sample,
databaseDir = dir.tmp,
internalDir = FALSE,
matrices = NULL,
```

```
normality = NULL,
dampingFactor = 0.85,
niter = 10)
## Load database
myFELLA.DATA <- loadKEGGdata(
dir.tmp,
internalDir = FALSE)
myFELLA.DATA

######################

## Not run:
## Full example

## First step: graph for Mus musculus discarding the mmu01100 pathway
## (an analog example can be built from human using organism = "hsa")
g.mmu <- buildGraphFromKEGGREST(
organism = "mmu",
filter.path = "mmu01100")
summary(g.mmu)
cat(comment(g.mmu))

## Second step: build internal files for this graph
## (consumes some time and memory, especially if we compute
"diffusion" and "pagerank" matrices)
buildDataFromGraph(
keggdata.graph = g.mmu,
databaseDir = "example_db_mmu",
internalDir = TRUE,
matrices = c("hypergeom", "diffusion", "pagerank"),
normality = c("diffusion", "pagerank"),
dampingFactor = 0.85,
niter = 1e3)
## Third step: load the internal files into a FELLA.DATA object
FELLA.DATA.mmu <- loadKEGGdata(
"example_db_mmu",
internalDir = TRUE,
loadMatrix = c("diffusion", "pagerank"))
FELLA.DATA.mmu

## End(Not run)
```

---

enrich-funs *Functions to map and enrich a list of metabolites*

Description

Function `defineCompounds` creates a `FELLA.USER` object from a list of compounds and a `FELLA.DATA` object.

Functions runHypergeom, runDiffusion and runPagerank perform an enrichment on a FELLA.USER with the mapped input metabolites (through defineCompounds) and a FELLA.DATA object. They are based on the hypergeometric test, the heat diffusion model and the PageRank algorithm, respectively.

Function enrich is a wrapper with the following order: loadKEGGdata (optional), defineCompounds and one or more in runHypergeom, runDiffusion and runPagerank

Usage

```
defineCompounds(compounds = NULL, compoundsBackground = NULL,
    data = NULL)

runHypergeom(object = NULL, data = NULL, p.adjust = "fdr")

runDiffusion(object = NULL, data = NULL, approx = "normality",
    t.df = 10, niter = 1000)

runPagerank(object = NULL, data = NULL, approx = "normality",
    dampingFactor = 0.85, t.df = 10, niter = 1000)

enrich(compounds = NULL, compoundsBackground = NULL,
    methods = listMethods(), loadMatrix = "none", approx = "normality",
    t.df = 10, niter = 1000, databaseDir = NULL, internalDir = TRUE,
    data = NULL, ...)
```

Arguments

compounds	Character vector containing the KEGG IDs of the compounds considered as affected
compoundsBackground	Character vector containing the KEGG IDs of the compounds that belong to the background. Can be NULL for the default background (all compounds)
data	FELLA.DATA object
object	FELLA.USER object
p.adjust	Character passed to the p.adjust method
approx	Character: "simulation" for Monte Carlo, "normality", "gamma" or "t" for parametric approaches
t.df	Numeric value; number of degrees of freedom of the t distribution if the approximation approx = "t" is used
niter	Number of iterations (permutations) for Monte Carlo ("simulation"), must be a numeric value between 1e2 and 1e5
dampingFactor	Numeric value between 0 and 1 (none inclusive), damping factor d for PageRank (page.rank)
methods	Character vector, containing some of: "hypergeom", "diffusion", "pagerank"
loadMatrix	Character vector to choose if heavy matrices should be loaded. Can contain: "diffusion", "pagerank"

databaseDir	Character, path to load the FELLA.DATA object if it is not already passed through the argument data
internalDir	Logical, is the directory located in the package directory?
...	Further arguments for the enrichment function(s) runDiffusion, runPagerank

Details

Function defineCompounds maps the specficied list of KEGG compounds [Kanehisa, 2017], usually from an experimental metabolomics study, to the graph contained in the FELLA.DATA object. Importantly, the names must be KEGG ids, so other formats (common names, HMDB ids, etc) must be mapped to KEGG first. For example, through the "Compound ID Conversion" tool in MetaboAnalyst [Xia, 2015]. The user can also define a personalised background as a list of KEGG compound ids, which should be more extensive than the list of input metabolites. Once the compounds are mapped, the enrichment can be performed through runHypergeom, runDiffusion and runPagerank.

Function runHypergeom performs an over representation analysis through the hypergeometric test [Fisher, 1935] on a FELLA.USER object with mapped metabolites and a FELLA.DATA object. If a custom background was specified, it will be used. This approach is included for completeness and it is not the main purpose behind the FELLA package. Importantly, runHypergeom is not a hypergeometric test using the original KEGG pathways. Instead, a compound "belongs" to a "pathway" if it can reach the original pathway in the upwards-directed KEGG graph. This is a way to evaluate enrichment including indirect connections to a pathway, e.g. through an enzymatic family. New "pathways" are expected to be larger than the original pathways in this analysis and therefore the results can differ from the standard over representation.

Function runDiffusion performs the diffusion-based enrichment on a FELLA.USER object with mapped metabolites and a FELLA.DATA object [Picart-Armada, 2017]. If a custom background was specified, it will be used. The idea behind the heat diffusion is the usage of the finite difference formulation of the heat equation to propagate labels from the metabolites to the rest of the graph.

Following the notation in [Picart-Armada, 2017], the temperatures (diffusion scores) are computed as:

$$T = -KI^{-1} \cdot G$$

G is an indicator vector of the input metabolites (1 if input metabolite, 0 otherwise). KI is the matrix -KI = L + B, being L the unnormalised graph Laplacian and B the diagonal matrix with B[i,i] = 1 if node i is a pathway and B[i,i] = 0 otherwise.

Equivalently, with the notation in the HotNet approach [Vandin, 2011], the stationary temperature is named fs:

$$f^s = L_\gamma^{-1} \cdot b^s$$

bs is the indicator vector G from above. Lgamma, on the other hand, is found as Lgamma = L + gamma*I, where L is the unnormalised graph Laplacian, gamma is the first order leaking rate and I is the identity matrix. In our formulation, only the pathway nodes are allowed to leak, therefore I is switched to B. The parameter gamma is set to gamma = 1.

The input metabolites are forced to stay warm, propagating flow to all the nodes in the network. However, only pathway nodes are allowed to evacuate this flow, so that its directionality is bottom-up. Further details on the setup of the diffusion process can be found in the supplementary file S2 from [Picart-Armada, 2017].

Finally, the warmest nodes in the graph are reported as the relevant sub-network. This will probably include some input metabolites and also reactions, enzymes, modules and pathways. Other metabolites can be suggested as well.

Function `runPagerank` performs the random walk based enrichment on a `FELLA.USER` object with mapped metabolites and a `FELLA.DATA` object. If a custom background was specified, it will be used. PageRank was originally conceived as a scoring system for websites [Page, 1999]. Intuitively, PageRank favours nodes that (1) have a large amount of nodes pointing at them, and (2) whose pointing nodes also have high scores. Classical PageRank is formulated in terms of a random walker - the PageRank of a given node is the stationary probability of the walker visiting it.

The walker chooses, in each step, whether to continue the random walk with probability `dampingFactor` or to restart it with probability `1 - dampingFactor`. In the original publication, `dampingFactor = 0.85`, which is the value used in `FELLA` by default. If he or she continues, an edge is picked from the outgoing edges in the current node with a probability proportional to its weight. If he or she restarts it, a node is uniformly picked from the whole graph. The "personalised PageRank" variant allows a user-defined distribution as the source of new random walks. The R package `igraph` contains such variant in its `page.rank` function [Csardi, 2006].

As described in the supplement S3 from [Picart-Armada, 2017], the PageRank `PR` can be computed as a column vector by imposing a stationary state in the probability. With a damping factor `d` and the user-defined distribution `p` as a column vector:

$$\mathrm{PR} = d \cdot M \cdot \mathrm{PR} + (1 - d) \cdot p$$

`M` is the matrix whose element `M[i,j]` is the probability of transitioning from `j` to `i`. If node `j` has outgoing edges, their probability is proportional to their weight - all weights must be positive. If node `j` has no outgoing edges, the probability is uniform over all the nodes, i.e. `M[i,j] = 1/nrow(M)` for every `i`. Note that all the columns from `M` sum up exactly `1`. This leads to an expression to compute PageRank:

$$\mathrm{PR} = (1 - d)p \cdot (I - dM)^{-1}$$

The idea behind the method `"pagerank"` is closely related to `"diffusion"`. Relevant metabolites are the sources of new random walks and nodes are scored through their PageRank. Specifically, `p` is set to a uniform probability on the input metabolites. More details on the setup can be found in the supplementary file S3 from [Picart-Armada, 2017].

There is an important detail for `"diffusion"` and `"pagerank"`: the scores are statistically normalised. Omitting this normalisation leads to a systematic bias, especially in pathway nodes, as described in [Picart-Armada, 2017].

Therefore, in both cases, scores undergo a normalisation through permutation analysis. The score of a node `i` is compared to its null distribution under input permutation, leading to their p-scores. As described in [Picart-Armada, 2017], two alternatives are offered: a parametric and deterministic approach and a non-parametric, stochastic one.

Stochastic Monte Carlo trials (`"simulation"`) imply randomly permuting the input `niter` times and counting, for each node `i`, how many trials led to an equally or more extreme value than the original score. An empirical p-value is returned [North, 2002].

On the other hand, the parametric scores (`approx = "normality"`) give a z-score for such permutation analysis. The expected value and variance of such null distributions are known quantities, see supplementary file S4 from [Picart-Armada, 2017]. To work in the same range `[0,1]`, z-scores are transformed using the routine `pnorm`. The user can also choose the Student's t using `approx = "t"` and choosing a number of degrees of freedom through `t.df`. This uses the function `pt` instead. Alternatively, a gamma distribution can be used by setting `approx = "gamma"`. The theoretical mean (E) and variance (V) are used to define the shape (E^2/V) and scale (V/E) of the gamma distribution, and `pgamma` to map to [0,1].

Any sub-network prioritised by `"diffusion"` and `"pagerank"` is selected by applying a threshold on the p-scores.

Finally, the function `enrich` is a wrapper to perform the enrichment analysis. If no `FELLA.DATA` object is supplied, it loads it, maps the affected compounds and performs the desired enrichment(s) with a single call. Returned is a list with the loaded `FELLA.DATA` object and the results in a `FELLA.USER` object. Conversely, the user can supply the `FELLA.DATA` object and the wrapper will map the metabolites and run the desired enrichment method(s). In this case, only the `FELLA.USER` will be returned.

Value

`defineCompounds` returns the `FELLA.USER` object with the mapped metabolites, ready to be enriched.

`runHypergeom` returns a `FELLA.USER` object updated with the hypergeometric test results

`runDiffusion` returns a `FELLA.USER` object updated with the diffusion enrichment results

`runPagerank` returns a `FELLA.USER` object updated with the PageRank enrichment results

`enrich` returns a `FELLA.USER` object updated with the desired enrichment results if the `FELLA.DATA` was supplied. Otherwise, a list with the freshly loaded `FELLA.DATA` object and the corresponding enrichment in the `FELLA.USER` object.

References

Kanehisa, M., Furumichi, M., Tanabe, M., Sato, Y., & Morishima, K. (2017). KEGG: new perspectives on genomes, pathways, diseases and drugs. Nucleic acids research, 45(D1), D353-D361.

Xia, J., Sinelnikov, I. V., Han, B., & Wishart, D. S. (2015). MetaboAnalyst 3.0 - making metabolomics more meaningful. Nucleic acids research, 43(W1), W251-W257.

Fisher, R. A. (1935). The logic of inductive inference. Journal of the Royal Statistical Society, 98(1), 39-82.

Picart-Armada, S., Fernandez-Albert, F., Vinaixa, M., Rodriguez, M. A., Aivio, S., Stracker, T. H., Yanes, O., & Perera-Lluna, A. (2017). Null diffusion-based enrichment for metabolomics data. PLOS ONE, 12(12), e0189012.

Vandin, F., Upfal, E., & Raphael, B. J. (2011). Algorithms for detecting significantly mutated pathways in cancer. Journal of Computational Biology, 18(3), 507-522.

Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). The PageRank citation ranking: Bringing order to the web. Stanford InfoLab.

Csardi, G., & Nepusz, T. (2006). The igraph software package for complex network research. InterJournal, Complex Systems, 1695(5), 1-9.

North, B. V., Curtis, D., & Sham, P. C. (2002). A note on the calculation of empirical P values from Monte Carlo procedures. American journal of human genetics, 71(2), 439.

Examples

```
## Load the internal database.
## This one is a toy example!
## Do not use as a regular database
data(FELLA.sample)
## Load a list of compounds to enrich
data(input.sample)

######################
## Example, step by step

## First, map the compounds
obj <- defineCompounds(
compounds = c(input.sample, "I_dont_map", "me_neither"),
data = FELLA.sample)
obj
## See the mapped and unmapped compounds
getInput(obj)
getExcluded(obj)
## Compounds are already mapped
## We can enrich using any method now

## If no compounds are mapped an error is thrown. Example:
## Not run:
data(FELLA.sample)
obj <- defineCompounds(
compounds = c("C00049", "C00050"),
data = FELLA.sample)
## End(Not run)

## Enrich using hypergeometric test
obj <- runHypergeom(
object = obj,
data = FELLA.sample)
obj

## Enrich using diffusion
## Note how the results are added;
## the hypergeometric results are not overwritten
obj <- runDiffusion(
object = obj,
approx = "normality",
data = FELLA.sample)
obj
```

```
## Enrich using PageRank
## Again, this does not overwrite other methods
obj <- runPagerank(
object = obj,
approx = "simulation",
data = FELLA.sample)
obj

######################
## Example using the "enrich" wrapper

## Only diffusion
obj.wrap <- enrich(
compounds = input.sample,
method = "diffusion",
data = FELLA.sample)
obj.wrap

## All the methods
obj.wrap <- enrich(
compounds = input.sample,
methods = FELLA::listMethods(),
data = FELLA.sample)
obj.wrap
```

`export-funs` *Generate and manipulate tables and sub-networks from an enrichment*

Description

In general, `generateResultsTable`, `generateEnzymesTable` and `generateResultsGraph` provide the results of an enrichment in several formats.

Function `generateResultsTable` returns a table that contains the best hits from a `FELLA.USER` object with a successful enrichment analysis. Similarly, `generateEnzymesTable` returns a data frame with the best scoring enzyme families and their annotated genes.

Function `generateResultsGraph` gives a sub-network, plottable through `plotGraph`, witht the nodes with the lowest `p.score` from an enrichment analysis. Function `addGOToGraph` can be applied to such sub-networks to overlay GO labels and similarity to a user-defined GO term.

Function `exportResults` is a wrapper around `generateResultsTable`, `generateEnzymesTable` and `generateResultsGraph` to write the results to files.

Usage

```
generateResultsTable(method = "diffusion", threshold = 0.05,
    plimit = 15, nlimit = 250, LabelLengthAtPlot = 45,
    capPscores = 1e-06, object = NULL, data = NULL, ...)
```

```
generateEnzymesTable(method = "diffusion", threshold = 0.05,
    nlimit = 250, LabelLengthAtPlot = 45, capPscores = 1e-06,
    mart.options = list(biomart = "ensembl", dataset =
    "hsapiens_gene_ensembl"), object = NULL, data = NULL, ...)

generateResultsGraph(method = "diffusion", threshold = 0.05,
    plimit = 15, nlimit = 250, thresholdConnectedComponent = 0.05,
    LabelLengthAtPlot = 22, object = NULL, data = NULL, ...)

exportResults(format = "csv", file = "myOutput",
    method = "diffusion", object = NULL, data = NULL, ...)

addGOToGraph(graph = NULL, GOterm = NULL, godata.options = list(OrgDb
    = "org.Hs.eg.db", ont = "CC"), mart.options = list(biomart = "ensembl",
    dataset = "hsapiens_gene_ensembl"))

plotGraph(graph = NULL, layout = FALSE, graph.layout = NULL,
    plotLegend = TRUE, plot.fun = "plot.igraph", NamesAsLabels = TRUE,
    ...)
```

Arguments

`method`	one in `"diffusion"`, `"pagerank"`
`threshold`	Numeric value between 0 and 1. `p.score` threshold applied when filtering KEGG nodes. Lower thresholds are more stringent.
`plimit`	Pathway limit, must be a numeric value between 1 and 50. Limits the amount of pathways in `method = "hypergeom"`
`nlimit`	Node limit, must be a numeric value between 1 and 1000. Limits the order of the solution sub-graph when in `method = "diffusion"` and `method = "pagerank"`
`LabelLengthAtPlot`	Numeric value between 10 and 50. Maximum length that a label can reach when plotting the graph. The remaining characters will be truncated using "..."
`capPscores`	Numeric value, minimum p-score admitted for the readable formatting. Smaller p-scores will be displayed as < `capPscores`
`object`	FELLA.USER object
`data`	FELLA.DATA object
`...`	Optional arguments for the plotting function in `plotGraph`. Arguments passed to the exporting function in `exportResults`. Ignored otherwise.
`mart.options`	List, options for the `biomaRt` function `getBM`. Importantly, this defines the organism, see `listDatasets` for possibilities. If calling `generateEnzymesTable`, the user can set `mart.options = NULL` to avoid adding GO labels to enzymes.
`thresholdConnectedComponent`	Numeric value between 0 and 1. Connected components that are below the threshold are kept, while the ones exceeding it (because they are too small) are discarded.

format	Character, one of: "csv" for regular results table, "enzyme" for table with enzyme data, "igraph" for igraph format. Alternatively, any format supported by igraph, see write_graph
file	Character specifying the output file name
graph	An **igraph** object, typically a small one, coming from an enrichment through "diffusion" or "pagerank".
GOterm	Character, GO entry to draw semantic similarity in the solution graph. If NULL, the GO labels will be appended without similarities.
godata.options	List, options for the database creator godata
layout	Logical, should the plot be returned as a layout?
graph.layout	Two-column numeric matrix, if this argument is not null then it is used as graph layout
plotLegend	Logical, should the legend be plotted as well?
plot.fun	Character, can be either plot.igraph or tkplot
NamesAsLabels	Logical, should KEGG names be displayed as labels instead of KEGG identifiers?

Details

Functions generateResultsTable and generateEnzymesTable need a FELLA.DATA object and a FELLA.USER object with a successful enrichment. generateResultsTable provides the entries whose p-score is below the chosen threshold in a tabular format. generateEnzymesTable returns a table that contains (1) the enzymes that are below the user-defined p-score threshold, along with (2) the genes that belong to the enzymatic families in the organism defined in the database, and (3) GO labels of such enzymes, if mart.options is not NULL and points to the right database.

Function generateResultsGraph returns an **igraph** object with a relevant sub-network for manual examination. A FELLA.USER object with a successful enrichment analysis and the corresponding FELLA.DATA must be supplied. Graph nodes are prioritised by p.score and selected through the most stringent between (1) p.score threshold and (2) maximum number of nodes nlimit.

There is an additional filtering feature for tiny connected components, controllable through thresholdConnectedComponent (smaller is stricter). The user can choose to turn off this filter by setting thresholdConnectedComponent = 1. The idea is to discard connected components so small that are likely to arise from random selection of nodes. Let k be the order of the current sub-network. A connected component of order r will be kept only if the probability that a random subgraph from the whole KEGG knowledge model of order k contains a connected component of order at least r is smaller than thresholdConnectedComponent. Such probabilities are estimated during buildDataFromGraph; the amount of random trials can be controlled by its niter argument.

Function exportResults writes the enrichment results as the specified filetype. Options are: a csv table ("csv"), an enzyme csv table ("enzyme") an **igraph** object as an RData file, or any format supported by igraph's write_graph.

Function addGOToGraph takes and returns a graph object with class **igraph** adding the following attributes: GO labels in V(graph)$GO, and semantic similarities in V(graph)$GO.simil if GOterm != NULL.

The GO database describes genes in terms of three ontologies: molecular function (MF), biological process (BP) and cellular component (CC) [Gene Ontology Consortium, 2015]. The user can be interested in finding which enzymatic families reported with a low `p.score` are closest to a particular GO term. To assess similarity between GO labels, FELLA uses the semantic similarity defined in [Yu, 2010] and their implementation in the **GOSemSim** R package. The user will obtain, for each enzymatic family, the closest GO term to his or her GO query and the semantic similarity between them. Exact matches have a similarity of `1`. Function `plotGraph` detects the presence of the GO similarity option and plots its magnitude.

Function `plotGraph` plots a solution graph from the diffusion and pagerank analysis. For plotting hypergeom results, please use `plot` instead. Specific colors and shapes for each KEGG category are used: pathways are maroon, modules are violet, enzymes are orange, reactions are blue and compounds are green. If the graph contains the similarity to a GO term, enzymes will be displayed as triangles whose color depicts the strength of such measure (yellow: weak, purple: strong). At the moment, `plotGraph` allows plotting throug the static `plot.igraph` and the interactive `tkplot`.

Value

`generateResultsTable` returns a data.frame that contains the nodes below the `p.score` threshold from an enrichment analysis

`generateEnzymesTable` returns a data.frame that contains the enzymes below the `p.score` threshold, along with their genes and GO labels

`generateResultsGraph` returns an **igraph** object: a sub-network from the whole KEGG knowledge model under the specified thresholds (`threshold` and `thresholdConnectedComponent`)

`exportResults` returns `invisible()`, but as a side effect the specified `file` is created.

`addGOToGraph` returns an **igraph** object, which is the input `graph` with extra attributes: GO labels in `V(graph)$GO`, and semantic similarities in `V(graph)$GO.simil` if `GOterm != NULL`

`plotGraph` returns `invisible()` if `layout = F` and the plotting layout as a data.frame otherwise.

References

Gene Ontology Consortium. (2015). Gene ontology consortium: going forward. Nucleic acids research, 43(D1), D1049-D1056.

Yu, G., Li, F., Qin, Y., Bo, X., Wu, Y., & Wang, S. (2010). GOSemSim: an R package for measuring semantic similarity among GO terms and gene products. Bioinformatics, 26(7), 976-978.

Examples

```
## First generate a toy enrichment
library(igraph)
data(FELLA.sample)
data(input.sample)
## Enrich input
obj <- enrich(
compounds = input.sample,
data = FELLA.sample)

######################
## Results table
```

```
tab.res <- generateResultsTable(
method = "hypergeom",
threshold = 0.1,
object = obj,
data = FELLA.sample)
head(tab.res)

tab.res <- generateResultsTable(
method = "diffusion",
threshold = 0.1,
object = obj,
data = FELLA.sample)
head(tab.res)

######################
## Use wrapper to write the table to a file
out.file <- tempfile()
exportResults(
format = "csv",
threshold = 0.1,
file = out.file,
object = obj,
data = FELLA.sample)
tab.wrap <- read.csv(out.file)
head(tab.wrap)

######################
## Enzymes table
tab.ec <- generateEnzymesTable(
threshold = 0.1,
object = obj,
data = FELLA.sample,
mart.options = NULL)
head(tab.ec)

######################
## Generate graph
g.res <- generateResultsGraph(
method = "pagerank",
threshold = 0.1,
object = obj,
data = FELLA.sample)
g.res

## Plot graph (without GO terms)
plotGraph(g.res)

## Add similarity to the GO CC term "mitochondrion"
## Not run:
g.cc <- FELLA:::addGOToGraph(
graph = g.res,
GOterm = "GO:0005739")
```

```
## Plot graph (with GO terms)
plotGraph(g.cc)

## Without the CC
any(V(g.res)$GO.simil >= 0)
## With the CC
v.cc <- unlist(V(g.cc)$GO.simil)
sum(v.cc >= 0, na.rm = TRUE)
## Similarity values
table(v.cc)

## End(Not run)
```

FELLA *The FELLA package*

Description

FELLA is a metabolomics data enrichment tool that contextualises a list of metabolites using KEGG reactions, enzymes, modules and pathways [Picart-Armada, 2017].

Details

FELLA can build knowledge models for the desired organism from the KEGG database [Kanehisa, 2017]. Once a model is ready, the input for the enrichment is introduced as a list of affected metabolites (as KEGG IDs). The output contains a comprehensive biological network layout that relates relevant pathways to the affected metabolites. Results are available in network and tabular format.

FELLA is equipped with a simple graphical interface for the lay user, deployed through launchApp.

FELLA relies mainly on the following packages: KEGGREST for the queries to the KEGG server [Tenenbaum, 2013], **igraph** for the network support [Csardi, 2006] and **shiny** for the graphical user interface [Chang, 2017].

References

Methodology:

Picart-Armada, S., Fernandez-Albert, F., Vinaixa, M., Rodriguez, M. A., Aivio, S., Stracker, T. H., Yanes, O., & Perera-Lluna, A. (2017). Null diffusion-based enrichment for metabolomics data. PLOS ONE, 12(12), e0189012.

Database:

Kanehisa, M., Furumichi, M., Tanabe, M., Sato, Y., & Morishima, K. (2017). KEGG: new perspectives on genomes, pathways, diseases and drugs. Nucleic acids research, 45(D1), D353-D361.

Main dependencies:

Tenenbaum, D. (2013). KEGGREST: Client-side REST access to KEGG. R package version, 1(1).

Csardi, G., & Nepusz, T. (2006). The igraph software package for complex network research. InterJournal, Complex Systems, 1695(5), 1-9.

Chang, W., Cheng, J., Allaire, JJ., Xie, Y., & McPherson, J. (2017). shiny: Web Application Framework for R. R package version 1.0.5. https://CRAN.R-project.org/package=shiny

Examples

```
## Walkthrough
browseVignettes("FELLA")
## I: create database
?buildGraphFromKEGGREST
## II: enrich data
?enrich
## III: export results
?exportResults
```

FELLA.DATA-class *An S4 class to represent all the necessary KEGG data*

Description

An S4 class to represent all the necessary KEGG data

"show" is an S4 method to show a FELLA.DATA object

Usage

```
## S4 method for signature 'FELLA.DATA'
show(object)
```

Arguments

object	A FELLA.DATA object

Value

show returns invisible()

Slots

keggdata A D.keggdata S4 object

hypergeom A D.hypergeom S4 object

diffusion A D.diffusion S4 object

pagerank A D.pagerank S4 object

FELLA.sample *FELLA.DATA sample data*

Description

This FELLA.DATA object is a small KEGG graph object. Despite being a small database that only contains the two metabolic pathways hsa00010 - Glycolysis / Gluconeogenesis, and hsa00640 - Propanoate metabolism, it is useful to play around with FELLA's functions. It is also used for internal testing of this package.

Usage

```
data(FELLA.sample)
```

Format

An object of class FELLA.DATA of length 1.

Value

A FELLA.DATA object

Source

Generated from a mid-2017 KEGG release (http://www.genome.jp/kegg/)

Examples

```
data(FELLA.sample)
```

FELLA.USER-class *An S4 class to save all the user analysis data*

Description

Assigning the value of show to a variable will provide small data frames with the best scoring pathways (hypergeom) and the best nodes in the KEGG network (diffusion and pagerank)

Usage

```
## S4 method for signature 'FELLA.USER'
show(object)

## S4 method for signature 'FELLA.USER,missing'
plot(x = new("FELLA.USER"),
    method = "hypergeom", threshold = 0.05, plimit = 15,
    nlimit = 250, layout = FALSE, thresholdConnectedComponent = 0.05,
    LabelLengthAtPlot = 22, data = NULL, ...)
```

Arguments

object	A FELLA.USER object
x	A FELLA.USER object
method	Character, exactly one of: "hypergeom", "diffusion", "pagerank"
threshold	Numeric value between 0 and 1. p.score threshold applied when filtering KEGG nodes. Lower thresholds are more stringent.
plimit	Pathway limit, must be a numeric value between 1 and 50. Limits the amount of pathways in method = "hypergeom"
nlimit	Node limit, must be a numeric value between 1 and 1000. Limits the order of the solution sub-graph when in method = "diffusion" and method = "pagerank"
layout	Logical, should the plot be returned as a layout?
thresholdConnectedComponent	Numeric value between 0 and 1. Connected components that are below the threshold are kept, while the ones exceeding it (because they are too small) are discarded.
LabelLengthAtPlot	Numeric value between 10 and 50. Maximum length that a label can reach when plotting the graph. The remaining characters will be truncated using "..."
data	FELLA.DATA object
...	Additional arguments passed to plotting functions

Value

show invisibly returns a list of data frames with the best hits for each applied method

plot returns a layout if layout = T, otherwise invisible()

Slots

userinput A U.userinput S4 object

hypergeom A U.hypergeom S4 object

diffusion A U.diffusion S4 object

pagerank A U.pagerank S4 object

---

getBackground *Get compounds in the defined background*

Description

Extractor function for the compounds defined as background

Usage

```
getBackground(object)
```

Arguments

object	FELLA.USER object

Value

Vector of compounds in the background. If this vector is empty, all the compounds are used as background by default.

Examples

```
data(FELLA.sample)
data(input.sample)
input <- head(input.sample, 12)

## If the background is default, we see an empty vector
## Note that the number of iterations is really small in the example
obj <- enrich(
compounds = input,
method = "diffusion",
approx = "simulation",
niter = 100,
data = FELLA.sample)

getBackground(obj)

## Otherwise we see the background compounds that mapped to the graph
obj <- enrich(
compounds = input,
compoundsBackground = input.sample,
method = "diffusion",
approx = "simulation",
niter = 100,
data = FELLA.sample)
getBackground(obj)
```

---

getCom *Get community*

Description

Extractor function for all the nodes from a level/community of KEGG graph

Usage

```
getCom(data, level, format = "name")
```

Arguments

data	FELLA.DATA object
level	Desired level, can be coded as a number or a character: 1 or "pathway"; 2 or "module"; 3 or "enzyme"; 4 or "reaction"; 5 or "compound".
format	Format of the output, "name" returns KEGG IDs whereas "id" returns vertices IDs

Value

Vector of the names/ids of the desired KEGG graph community

Examples

```
data(FELLA.sample)
## Pathways
getCom(FELLA.sample, 1, format = "name")
getCom(FELLA.sample, 1, format = "id")
## Modules
getCom(FELLA.sample, 2)
## Enzymes
head(getCom(FELLA.sample, 3))
## Reactions
head(getCom(FELLA.sample, 4))
## Compounds
head(getCom(FELLA.sample, 5))
```

---

getExcluded *Get excluded compounds*

Description

Extractor function for the compounds in the input that were not mapped to the KEGG graph

Usage

```
getExcluded(object)
```

Arguments

object	FELLA.USER object

Value

Vector of the excluded compounds

Examples

```
data(FELLA.sample)
data(input.sample)

## No excluded compounds
obj <- defineCompounds(
compounds = input.sample,
data = FELLA.sample)
getExcluded(obj)

## One compound does not map
## The user gets a warning as well
obj <- defineCompounds(
compounds = c(input.sample, "intruder"),
data = FELLA.sample)
getExcluded(obj)
```

---

getGraph *Get KEGG graph*

Description

Extractor function for the KEGG graph from the FELLA.DATA object

Usage

```
getGraph(data)
```

Arguments

data	FELLA.DATA object

Value

KEGG graph as an **igraph** object

Examples

```
data(FELLA.sample)
g <- getGraph(FELLA.sample)
class(g)
```

getInfo *Get KEGG version info*

Description

Extractor function for the info about the KEGG version used to build the FELLA.DATA object

Usage

```
getInfo(data)
```

Arguments

data	FELLA.DATA object

Value

Character containing the KEGG release details

Examples

```
data(FELLA.sample)
getInfo(FELLA.sample)
```

getInput *Get metabolites in the input*

Description

Extractor function for the metabolites specified by the user in the input

Usage

```
getInput(object)
```

Arguments

object	FELLA.USER object

Value

Vector of metabolites in the input

Examples

```
data(FELLA.sample)
data(input.sample)

## No excluded compounds: the input is recovered as is
obj <- defineCompounds(
compounds = input.sample,
data = FELLA.sample)
i1 <- getInput(obj)

## One compound does not map: the input contains only the mapped entities
obj <- defineCompounds(
compounds = c(input.sample, "intruder"),
data = FELLA.sample)
i2 <- getInput(obj)

identical(sort(i1), sort(i2))
```

`getMatrix` *Get matrix for the desired methodology*

Description

Extractor function for the matrices of hypergeometric, diffusion and PageRank methodologies

Usage

```
getMatrix(data, method)
```

Arguments

`data`	FELLA.DATA object
`method`	Character, exactly one of: `"hypergeom"`, `"diffusion"`, `"pagerank"`

Value

Matrix for the desired methodology (internal usage)

Examples

```
## This function is internal
attach(environment(FELLA:::getMatrix))
data(FELLA.sample)
# When a matrix is loaded:
x <- getMatrix(FELLA.sample, "hypergeom")
dim(x)
# When it is not:
y <- getMatrix(FELLA.sample, "diffusion")
dim(y)
y
```

getName *Map KEGG identifiers to KEGG names*

Description

Map KEGG identifiers to KEGG names, multiple names for an ID are reported if annotated. The KEGG identifiers may have mixed levels.

Usage

```
getName(data, id)
```

Arguments

data	FELLA.DATA object
id	KEGG IDs whose name is desired

Value

List whose names are KEGG IDs and whose entries are the vectors of matches

Examples

```
data(FELLA.sample)
getName(FELLA.sample, c("C00002", "C00040"))
```

getPscores *Get p-scores from the desired methodology*

Description

Extractor function for the p-scores using the desired methodology

Usage

```
getPscores(object, method)
```

Arguments

object	FELLA.USER object
method	Character, exactly one of: "hypergeom", "diffusion", "pagerank"

Value

Named vector of p-scores

Examples

```
data(FELLA.sample)
data(input.sample)
obj <- enrich(
compounds = input.sample,
data = FELLA.sample)
p <- getPscores(obj, "diffusion")
sum(p < 0.1)
```

`getStatus` *Get the slot "status"*

Description

Extractor function for the slot "status" for the KEGG data

Usage

```
getStatus(data)
```

Arguments

`data`	FELLA.DATA object

Value

Slot "status" (internal usage)

Examples

```
## This function is internal

data(FELLA.sample)

## Is the object loade?
FELLA:::getStatus(FELLA.sample)
FELLA:::getStatus(new("FELLA.DATA"))
```

getSums *Get rowSums/squaredRowSums*

Description

Extractor function for rowSums/squaredRowSums

Usage

```
getSums(data, method, squared)
```

Arguments

data	FELLA.DATA object
method	Character, exactly one of: "hypergeom", "diffusion", "pagerank"
squared	Logical, whether to return rowSums (F) or squaredRowSums (T)

Value

Named vector with rowSums/squaredRowSums (internal usage)

Examples

```
## This function is internal
attach(environment(FELLA:::getSums))
data(FELLA.sample)
rowsums <- getSums(FELLA.sample, "diffusion", squared = FALSE)
hist(rowsums)
```

getValid *Get the slot "valid"*

Description

Extractor function for the slot "valid"

Usage

```
getValid(object, method)
```

Arguments

object	FELLA.USER object
method	Character, exactly one of: "hypergeom", "diffusion", "pagerank"

Value

Slot "valid" (internal usage)

Examples

```
## This function is internal

data(FELLA.sample)
data(input.sample)

obj <- enrich(
compounds = input.sample,
method = "diffusion",
data = FELLA.sample)

## If the analysis is valid
FELLA:::getValid(obj, "diffusion")

## Otherwise
FELLA:::getValid(new("FELLA.USER"), "diffusion")
FELLA:::getValid(obj, "pagerank")
```

`input.sample` *A randomly generated list of affected metabolites*

Description

This `character` vector object has been generated using the sample data in the object `FELLA.sample`. The KEGG compounds have been chosen with preference for the hsa00640 pathway, so that the enrichment results choose pathway hsa00640 over hsa00010.

Usage

```
data(input.sample)
```

Format

An object of class `character` of length 30.

Value

A `character` vector containing 30 KEGG IDs

Source

Generated from a mid-2017 KEGG release (http://www.genome.jp/kegg/)

Examples

```
data(input.sample)
```

---

is.FELLA.DATA *Check FELLA.DATA class*

Description

Is x a FELLA.DATA object?

Usage

```
is.FELLA.DATA(x = NULL)
```

Arguments

x	Object to check

Value

Logical value stating if x is a FELLA.DATA object

Examples

```
data(FELLA.sample)
is.FELLA.DATA(FELLA.sample)
is.FELLA.DATA(42)
```

---

is.FELLA.USER *Check FELLA.USER class*

Description

Is x a FELLA.USER object?

Usage

```
is.FELLA.USER(x = NULL)
```

Arguments

x	Object to check

Value

Logical value stating if x is a FELLA.USER object

Examples

```
is.FELLA.USER(new("FELLA.USER"))
is.FELLA.USER(42)

data(FELLA.sample)
data(input.sample)
obj <- enrich(
compounds = input.sample,
method = "diffusion",
data = FELLA.sample)
is.FELLA.USER(obj)
```

---

launchApp *Launch a shiny app with FELLA*

Description

launchApp deploys a shiny application to perform the metabolomics data enrichment. Although this app does not provide all the options available in FELLA, it is easily accessible for the lay user.

Usage

```
launchApp(...)
```

Arguments

...	Parameters passed to runApp

Details

The graphical interface allows to: (1) upload the data and check if the KEGG ids have successfully mapped, (2) select database, set analysis and graphical parameters, (3) interactively browse the resulting sub-network as a graph or as a table, and (4) export such results as a table or a network. At least one database is needed before deploying the app. See ?buildDataFromGraph for further details.

Value

invisible(), but as a side effect the app will be launched

Examples

```
## Not run:
r <- try(launchApp())

## End(Not run)
```

listApprox *List of approximations*

Description

Available approximations for the analysis

Usage

```
listApprox()
```

Value

Character vector

Examples

```
listApprox()
```

listCategories *List of node categories*

Description

Node categories used in the internal representations

Usage

```
listCategories()
```

Value

Character vector

Examples

```
listCategories()
```

listInternalDatabases *List internal databases*

Description

This function lists the directories in the local database path

Usage

```
listInternalDatabases(full.names = FALSE)
```

Arguments

full.names	Logical, should full paths be returned?

Value

Vector with database directories

Examples

```
listInternalDatabases()
```

listMethods *List of methods*

Description

Methods available for the analysis

Usage

```
listMethods()
```

Value

Character vector

Examples

```
listMethods()
```

plotLegend *Internal function to add a legend to a graph plot*

Description

This function adds a legend to a solution plot. It can include the CC similarity.

Usage

```
plotLegend(GO.annot = FALSE, cex = 0.75)
```

Arguments

GO.annot	Logical, should GO annotations be included?
cex	Numeric value, cex parameter for the function legend

Value

This function is only used for its effect, so it returns invisible()

Examples

```
## This function is internal

library(igraph)
g <- barabasi.game(20)
plot(g)
FELLA:::plotLegend()
plot(g)
FELLA:::plotLegend(GO.annot = TRUE)
```

U.diffusion-class *An internal S4 class for the user data of the diffusion enrichment analysis*

Description

An internal S4 class for the user data of the diffusion enrichment analysis

Slots

valid Logical value; is the analysis valid?

pscores Named numeric vector with p-scores

approx Character; which approximation was used? Can be "simulation" for Monte Carlo; "normality", "gamma" or "t" for parametric approaches

niter Numeric value, number of iterations for the simulated approach

U.hypergeom-class *An internal S4 class for the user data of the hypergeometric over representation analysis*

Description

An internal S4 class for the user data of the hypergeometric over representation analysis

Slots

`valid` Logical value; is the analysis valid?

`pvalues` Named numeric vector with p-values

`pathhits` Numeric named vector with the quantities "sample_success" for the hypergeometric distribution (#affected in path)

`pathbackground` Numeric named vector with the quantities "total_success" for the hypergeometric distribution (total in path)

`nbackground` Numeric value, number of compoudns in the background. Equivalently, number of rows for the hypergeometric binary matrix

`ninput` Numeric value, number of affected compounds matched to the rownames

U.pagerank-class *An internal S4 class for the user data of the PageRank enrichment analysis*

Description

An internal S4 class for the user data of the PageRank enrichment analysis

Slots

`valid` Logical value; is the analysis valid?

`pscores` Named numeric vector with p-scores

`approx` Character; which approximation was used? Can be "simulation" for Monte Carlo; "normality", "gamma" or "t" for parametric approaches

`niter` Numeric value, number of iterations for the simulated approach

U.userinput-class *An internal S4 class for the user input data*

Description

An internal S4 class for the user input data

Slots

`metabolites` Character vector containing the affected compounds

`metabolitesbackground` Character vector containing the compounds for the personalised background. Optionally, can be `NULL` for default background

`excluded` Character vector containing the compounds that have been excluded because they cannot be mapped to KEGG graph compounds

Index

∗ **datasets**
FELLA.sample, 20
input.sample, 30

addGOToGraph *(*export-funs*)*, 13

buildDataFromGraph, *15*
buildDataFromGraph *(*data-funs*)*, 4
buildGraphFromKEGGREST *(*data-funs*)*, 4

D.diffusion-class, 3
D.hypergeom-class, 3
D.keggdata-class, 3
D.pagerank-class, 4
data-funs, 4
defineCompounds *(*enrich-funs*)*, 7

enrich *(*enrich-funs*)*, 7
enrich-funs, 7
export-funs, 13
exportResults *(*export-funs*)*, 13

FELLA, *5, 6*, *9*, 18, *32*
FELLA-package *(*FELLA*)*, 18
FELLA.DATA, *4*, *6–11*, *15*, *19, 20*, *31*
FELLA.DATA *(*FELLA.DATA-class*)*, 19
FELLA.DATA-class, 19
FELLA.sample, 20
FELLA.USER, *5*, *7–11*, *13*, *15*, *21*, *31*
FELLA.USER *(*FELLA.USER-class*)*, 20
FELLA.USER-class, 20

generateEnzymesTable *(*export-funs*)*, 13
generateResultsGraph *(*export-funs*)*, 13
generateResultsTable *(*export-funs*)*, 13
getBackground, 21
getBM, *14*
getCom, 22
getExcluded, 23
getGraph, 24
getInfo, 25
getInput, 25
getMatrix, 26
getName, 27
getPscores, 27
getStatus, 28
getSums, 29
getValid, 29
godata, *15*

input.sample, 30
is.FELLA.DATA, 31
is.FELLA.USER, 31

launchApp, *18*, 32, *32*
legend, *35*
listApprox, 33
listCategories, 33
listDatasets, *14*
listInternalDatabases, 34
listMethods, 34
loadKEGGdata *(*data-funs*)*, 4

p.adjust, *8*
page.rank, *5*, *8*, *10*
pgamma, *11*
plot,FELLA.USER,missing-method
*(*FELLA.USER-class*)*, 20
plot.igraph, *16*
plotGraph *(*export-funs*)*, 13
plotLegend, 35
pnorm, *11*
pt, *11*

runApp, *32*
runDiffusion *(*enrich-funs*)*, 7
runHypergeom *(*enrich-funs*)*, 7
runPagerank *(*enrich-funs*)*, 7

show,FELLA.DATA-method
*(*FELLA.DATA-class*)*, 19

show,FELLA.USER-method
(FELLA.USER-class), 20

tkplot, *16*

U.diffusion-class, 35
U.hypergeom-class, 36
U.pagerank-class, 36
U.userinput-class, 37

write_graph, *15*