Package ‘FELLA’

March 18, 2024

Type Package

Title Interpretation and enrichment for metabolomics data

Version 1.22.0

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

Enhances

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'
R topics documented:

.git_url https://git.bioconductor.org/packages/FELLA
.git_branch RELEASE_3_18
.git_last_commit 3444f45
.git_last_commit_date 2023-10-24
.Repository Bioconductor 3.18
.Date/Publication 2024-03-18
.Author Sergio Picart-Armada [aut, cre],
             Francesc Fernandez-Albert [aut],
             Alexandre Perera-Lluna [aut]

R topics documented:

.params .................................................. 3
.checkArguments ........................................ 5
.D.diffusion-class ..................................... 6
.D.hypergeom-class ..................................... 7
.D.keggdata-class ...................................... 7
.D.pagerank-class ...................................... 7
data-funs ................................................. 8
.enrich-funs ............................................. 11
.export-funs ............................................. 16
.FELLA .................................................... 21
.FELLA.DATA-class ...................................... 22
.FELLA.sample ........................................... 23
.FELLA.USER-class ...................................... 24
.getBackground ......................................... 25
.getCom .................................................... 26
.getExcluded ............................................. 27
.getGraph ............................................... 27
.getInfo .................................................. 28
.getInput ................................................. 29
.getMatrix ............................................... 29
.getName .................................................. 30
.getPscores .............................................. 31
.getPvaluesSize ....................................... 31
.getStatus ............................................... 32
.getSums ................................................... 33
.getValid ................................................. 33
.infer_con2ec .......................................... 33
.input.sample ............................................ 34
.is.FELLA.DATA ......................................... 36
.is.FELLA.USER ......................................... 36
.largestcc ............................................... 37
.launchApp ............................................... 38
.listApprox ............................................. 38
.listCategories ......................................... 39
.params

Description

This function eases parameter inheritance to centralise the documentation

Usage

.params()

Arguments

databaseDir  Path for the KEGG RData files
internalDir  Logical, is the directory located in the package directory?
object       FELLA.USER object
data          FELLA.DATA object
type          Character vector, containing entries in "hypergeom", "diffusion" or "pagerank"
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".
method        Character, exactly one of: "hypergeom", "diffusion", "pagerank"
methods       Character vector, containing some of: "hypergeom", "diffusion", "pagerank"
approx        Character: "simulation" for Monte Carlo, "normality", "gamma" or "t" for parametric approaches
loadMatrix    Character vector to choose if heavy matrices should be loaded. Can contain: "diffusion", "pagerank"
threshold     Numeric value between 0 and 1. p.score threshold applied when filtering KEGG nodes. Lower thresholds are more stringent.
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.
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"

niter  Number of iterations (permutations) for Monte Carlo ("simulation"), must be a numeric value between 1e2 and 1e5

layout  Logical, should the plot be returned as a layout?

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.

GONamesAsLabels  Logical, should GO names be displayed as labels instead of GO identifiers?

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 "...

godata.options  List, options for the database creator godata

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.

p.adjust  Character passed to the p.adjust method

dampingFactor  Numeric value between 0 and 1 (none inclusive), damping factor d for PageRank (page.rank)

t.df  Numeric value; number of degrees of freedom of the t distribution if the approximation approx = "t" is used

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)

NamesAsLabels  Logical, should KEGG names be displayed as labels instead of KEGG identifiers?

capPscores  Numeric value, minimum p-score admitted for the readable formatting. Smaller p-scores will be displayed as < capPscores

Value

NULL
checkArguments

Internal function to check arguments and give personalised errors

Description

This function checks if the arguments are of the desired type, length and range. If it fails, it gives an error explaining why the argument is invalid.

Usage

checkArguments(databaseDir = "myDatabase", internalDir = TRUE,
method = "diffusion", methods = "diffusion", approx = "normality",
loadMatrix = NULL, threshold = 0.05, plimit = 15, nlimit = 250,
niter = 1000, t.df = 10, dampingFactor = 0.85, layout = FALSE,
thresholdConnectedComponent = 0.05, GOterm = NULL,
GONamesAsLabels = TRUE, LabelLengthAtPlot = 22,
object = new("FELLA.USER"), data = new("FELLA.DATA"), ...)

Arguments

databaseDir Path for the KEGG RData files
internalDir Logical, is the directory located in the package directory?
method Character, exactly one of: "hypergeom", "diffusion", "pagerank"
methods Character vector, containing some of: "hypergeom", "diffusion", "pagerank"
approx Character: "simulation" for Monte Carlo, "normality", "gamma" or "t" for parametric approaches
loadMatrix Character vector to choose if heavy matrices should be loaded. Can contain: "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"
niter Number of iterations (permutations) for Monte Carlo ("simulation"), must be a numeric value between 1e2 and 1e5
t.df Numeric value; number of degrees of freedom of the t distribution if the approximation approx = "t" is used
dampingFactor Numeric value between 0 and 1 (non inclusive), damping factor d for PageRank (page.rank)
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.

GOterm
   Character, GO entry to draw semantic similarity in the solution graph. If NULL, the GO labels will be appended without similarities.

GONamesAsLabels
   Logical, should GO names be displayed as labels instead of GO identifiers?

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 "...

object
   FELLA.USER object

data
   FELLA.DATA object

... ignored arguments

Value

A list with values. Currently only a logical value named valid if the process runs smoothly. If the checking fails, it also returns an object called ans, which depends on the situation (can be the original object, NULL, et cetera).

Examples

## This function is internal
arg1 <- FELLA:::checkArguments(method = "hello")
arg1$valid
arg2 <- FELLA:::checkArguments(method = "diffusion")
arg2$valid

---

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**

```r
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 reminiscent 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 knowledge 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**


See Also
class FELLA.DATA

Examples

```r
## 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",
"example_db_mmu",
internalDir = TRUE,
loadMatrix = c("diffusion", "pagerank"))
FELLA.DATA.mmu

## End(Not run)

### 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**

```r
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 \( \text{approx} = "t" \) 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 specified 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:

\[ T = -K^{-1} \cdot G \]

\( G \) is an indicator vector of the input metabolites (1 if input metabolite, 0 otherwise). \( K \) is the matrix \(-K = L + B\), 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 \( f_s \):

\[ f_s = L^{-1} \cdot b^s \]

\( b_s \) is the indicator vector \( G \) from above. \( L_{\gamma} \), on the other hand, is found as \( L_{\gamma} = L + \gamma I \), 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 \( d \) or to restart it with probability \( 1 - d \). In the original publication, \( d = 0.85 \), 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:

\[ PR = d \cdot M \cdot PR + (1 - d) \cdot p \]
$M$ 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. $M[i,j] = 1/\text{row}(M)$ for every $i$. Note that all the columns from $M$ sum up exactly 1. This leads to an expression to compute PageRank:

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

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


Examples

```r
## 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
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`, with 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

```r
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 through 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.
Examples

```r
## 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)

# 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)
```

```r
## Enzymes table
```

References


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:

Database:

Main dependencies:

Examples

```r
## 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

```r
## 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

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)
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

```r
## 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()
**getBackground**

**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

---

**Description**

Extractor function for the compounds defined as background

**Usage**

```r
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**

```r
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",
```
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)
geExcluded(obj)

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

getGraph  

Get KEGG graph

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

Usage
getGraph(data)
getInfo

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

g 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

g 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)
g 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)

getPvaluesSize

Get matrix for the p-value regarding CC size

Description
Extractor function for the matrix containing p-value by CC size that compares to a random selection of nodes in the KEGG graph

Usage
getPvaluesSize(data)

Arguments

data FELLA.DATA object
### getStatus

**Value**

Matrix with p-values for CC size (internal usage)

**Examples**

```r
## This function is internal
attach(environment(FELLA:::getPvaluesSize))
data(FELLA.sample)
M <- getPvaluesSize(FELLA.sample)
dim(M)
summary(as.vector(M))
```

---

### getStatus

*Get the slot "status"

**Description**

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

**Usage**

```r
getStatus(data)
```

**Arguments**

- `data` FELLA.DATA object

**Value**

Slot "status" (internal usage)

**Examples**

```r
## This function is internal
data(FELLA.sample)

## Is the object loaded?
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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>FELLA.DATA object</td>
</tr>
<tr>
<td>method</td>
<td>Character, exactly one of: &quot;hypergeom&quot;, &quot;diffusion&quot;, &quot;pagerank&quot;</td>
</tr>
<tr>
<td>squared</td>
<td>Logical, whether to return rowSums (F) or squaredRowSums (T)</td>
</tr>
</tbody>
</table>

Value
Named vector with rowSums/squaredRowSums (internal usage)

Examples
```r
## 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
cgetValid(object, method)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>FELLA.USER object</td>
</tr>
<tr>
<td>method</td>
<td>Character, exactly one of: &quot;hypergeom&quot;, &quot;diffusion&quot;, &quot;pagerank&quot;</td>
</tr>
</tbody>
</table>
Value

Slot "valid" (internal usage)

Examples

```r
## 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")
```

### infere.con2ec

**Infer connections to EC**

Description

Function `infere.con2ec` infers network connections to KEGG EC families by passing through genes. This assumes that the category being mapped to enzymes is above them.

Usage

```r
infere.con2ec(ids, ent, ent2gene, gene2enyzme)
```

Arguments

- **ids**: Character vector of identifiers to map. For example, all the KEGG pathways
- **ent**: Character, entity that we are mapping (one of "pathway" and one of "module")
- **ent2gene**: Named character vector, names are the entity `ent` and values are genes
- **gene2enyzme**: Named character vector, names are genes and values are EC enzyme families category

Value

Two-column data frame. Column "from" contains the KEGG enzyme families whereas "to" contains the entity `ent`. 
Examples

```r
ids <- "hsa00010"
ent <- "pathway"
ent2gene <- c("hsa00010" = "hsa:10", "hsa00010" = "hsa:120")
gene2enzyme <- c("hsa:10" = "1.1.1.1", "hsa:120" = "1.2.3.4")
FELLA:::infere.con2ec(ids, ent, ent2gene, gene2enzyme)
```

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)

largestcc

Extract largest CC

Description

Function largestcc extracts the largest connected component of an igraph object

Usage

largestcc(graph)

Arguments

graph Igraph object

Value

Connected igraph object

Examples

library(igraph)
g <- barabasi.game(10) + graph.empty(10)
FELLA:::largestcc(g)
### 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

```r
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

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

Value
Character vector

Examples
listApprox()

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
mytriangle

Examples

```
listInternalDatabases()
```

---

listMethods  List of methods

---

**Description**

Methods available for the analysis

**Usage**

```
listMethods()
```

**Value**

Character vector

**Examples**

```
listMethods()
```

---

**mytriangle**  Add triangular shape to igraph plot

---

**Description**

This function enables the usage of triangles as shape in the function `plot.igraph`.

**Usage**

```
mytriangle(coords, v = NULL, params)
```

**Arguments**

- `coords`, `v`, `params`
  - clipping arguments, see `shapes`

**Value**

Plot symbols
plotBipartite

Examples

## This function is internal
library(igraph)

add.vertex.shape(
  "triangle", clip = shapes("circle")$clip,
  plot = FELLA:::mytriangle)

g <- barabasi.game(10)
plot(
g, vertex.shape = "triangle",
vertex.color = rainbow(vcount(g)),
vertex.size = seq(10, 20, length = vcount(g))
)

plotBipartite

Internal function to plot a bipartite graph

Description

This function plots a bipartite graph, tailored for the hypergeometric over representation analysis. As the nodes can only be compounds and pathways, the layout is simple and bipartite.

Usage

plotBipartite(graph = NULL, layout = FALSE, ...)

Arguments

graph Graph result that must come from the hypergeometric test analysis
layout Logical, should the plot be returned as a layout?
... Additional parameters passed to plot.igraph

Value

If layout = F then the value returned is invisible(). Otherwise, the layout is returned, also in an invisible fashion.

Examples

## This function is internal
data(FELLA.sample)
data(input.sample)
## Enrich input
obj <- enrich(
  compounds = input.sample,
data = FELLA.sample,
method = "hypergeom")
Generate the bipartite graph (only in the hypergeometric test)

```r
g <- generateResultsGraph(
  method = "hypergeom",
  threshold = 1,
  object = obj,
  data = FELLA.sample)
```

Plot it

```r
FELLA:::plotBipartite(g)
```

---

### 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**

```r
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**

```r
## This function is internal

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

Description
Sanitise KEGG identifiers

Usage
sanitise(x, category, organism)

Arguments
- **x**: Character vector, IDs to sanitise
- **category**: Character, one of: "pathway", "module", "enzyme", "ncbi", "reaction", "compound"

Value
Character vector, sanitised x

Examples
FELLA:::sanitise(c("path:hsa00010", "path:hsa00020"), "pathway", "hsa")

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 compounds 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-values
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
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, 23
  input.sample, 35

* internal
  .params, 3
  checkArguments, 5
  getPvaluesSize, 31
  infere.con2ec, 34
  largestcc, 37
  mytriangle, 40
  plotBipartite, 41
  sanitise, 43
  .params, 3
  addGOToGraph (export-funs), 16

buildDataFromGraph, 19
buildDataFromGraph (data-funs), 8
buildGraphFromKEGGREST (data-funs), 8
checkArguments, 5
D.diffusion-class, 6
D.hypergeom-class, 7
D.keggdata-class, 7
D.pagerank-class, 7
data-funs, 8
defineCompounds (enrich-funs), 11

enrich (enrich-funs), 11
enrich-funs, 11
export-funs, 16
exportResults (export-funs), 16

FELLA, 9, 12, 21, 38
FELLA-package (FELLA), 21
FELLA.DATA, 8–15, 18, 19, 23, 36
FELLA.DATA (FELLA.DATA-class), 22
FELLA.DATA-class, 22
FELLA.sample, 23
FELLA.USER, 9, 11–15, 17, 18, 24, 36
FELLA.USER (FELLA.USER-class), 24
FELLA.USER-class, 24

generateEnzymesTable (export-funs), 16
generateResultsGraph (export-funs), 16
generateResultsTable (export-funs), 16
getBackground, 25
getBM, 4, 18
getCom, 26
getExcluded, 27
getGraph, 27
getInfo, 28
getInput, 29
getMatrix, 29
getName, 30
getPscores, 31
getPvaluesSize, 31
getStatus, 32
getSums, 33
getValid, 33
godata, 4, 18

infere.con2ec, 34
input.sample, 35
is.FELLA.DATA, 36
is.FELLA.USER, 36
largestcc, 37
launchApp, 22, 38, 38
legend, 42
listApprox, 38
listCategories, 39
listDatasets, 4, 18
listInternalDatabases, 39
listMethods, 40
loadKEGGdata (data-funs), 8
mytriangle, 40

p.adjust, 4, 12
page.rank, 4, 5, 8, 12, 13
pgamma, 14
plot, FELLA.USER, missing-method
(FELLA.USER-class), 24
plot.igraph, 19, 40, 41
plotBipartite, 41
plotGraph (export-funs), 16
plotLegend, 42
pnorm, 14
pt, 14

runApp, 38
runDiffusion (enrich-funs), 11
runHypergeom (enrich-funs), 11
runPagerank (enrich-funs), 11

sanitise, 43
shapes, 40
show, FELLA.DATA-method
(FELLA.DATA-class), 22
show, FELLA.USER-method
(FELLA.USER-class), 24

tkplot, 19

U.diffusion-class, 43
U.hypergeom-class, 44
U.pagerank-class, 44
U.userinput-class, 45

write_graph, 18, 19