Package 'MetQy'

February 28, 2018

Title Metabolic Analysis using Queries

Version 1.1.0

BugReports `https://github.com/OSS-Lab/MetQy/issues`

Description MetQy facilitates analysis and data mining of KEGG. The package consists of several families of functions.'parseKEGG' functions allow for easy reading and formatting of the KEGG databases. 'query' functions carry out metabolic analysis given genes or genomes (both user-specified or from the KEGG database) and KEGG module search terms. 'plot' and 'analysis' functions facilitate data visualisation and analyses such as module variance and PCA. 'misc' functions support the other function families.

Depends R (>= 3.4.1)

Imports dplyr,ggplot2,gsubfn,reshape2,xtable

License University of Warwick non commercial use licence (see LICENCE file: https://github.com/OSS-Lab/MetQy/blob/master/LICENCE)

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

Suggests testthat

Author Andrea Martinez-Vernon [aut, cre],
Soyer Orkun [ctb],
Frederick Farrell [ctb]

Maintainer Andrea Martinez-Vernon <`asmvernon@gmail.com`>

R topics documented:

analysis_genomes_module_output

Process the output generated by `query_genomes_to_modules()`.

Description

Process the output generated by `query_genomes_to_modules()`.

Usage

```
analysis_genomes_module_output(FRACTION_MATRIX, QUERIES = NULL,
  outPath = "", report_file = "report.tex", figType = c(".eps", ".png"),
  byGenus = FALSE, bySpecies = FALSE, FACTOR = NULL, ...)
```

Arguments

`FRACTION_MATRIX`
- matrix. $MATRIX matrix generated by `query_genomes_to_modules()`.

`QUERIES` - optional. $QUERIES data frame generated by `query_genomes_to_modules()` containing the search query performed.

`outPath` - optional. String to indicate path to store output file and figures. Default (`Sys.Date()` as 'YYYY_MM_DD', e.g. '2000_01_31').

`report_file` - optional. File name for LaTex report (see Details for path handling). Default ("report.tex" saved to outPAth; use `NULL` to supress report generation).

`figType` - optional. Character vector to indicate file extension for figures. Default (c(".eps", ".png")).

`byGenus, bySpecies`
- optional, logical. Should a genus/species-based analysis be carried out? Default (FALSE).

`FACTOR` - optional. Character vector or list of character vectors indicating the grouping factor of the datasets. Default (NULL).

`...` - further arguments for analysis_pca_mean_distance_grouping or argument `factor_labs` for plot_scatter_byFactors.

Details

Below are the steps carried on the `query_genomes_to_modules()` output.

(1) Module fraction completeness (mfc) visualization of the `query_genomes_to_modules()` output;

(2) variance analysis to identify complete and absent modules, as well as modules with zero variance (i.e. same fraction present across all modules);

(3) a principal component analysis (PCA; using the `prcomp` from the 'stats' package) is done and a plot of the cumulative variance and the first to principal components are generated.

When `FACTOR` is specified, a mean distance is calculated as a proxy for inner-group spread (using as many dimensions as specified by the optional additional argument `nDim`; used by analysis_pca_mean_distance_grouping()).

Steps 1 and 2 can then be applied in a genus-based and species-based manner, when organism's names are given. In this instance, the average fraction for each module(s) is used to represent the genus or species, respectively.

It will be assumed that if `report_file` has a folder structure (identified by "\") a path has been given. Otherwise, the report will be written to `outPath`. If `report_file` is set to `NULL`, the report is not generated.

Value

This function automatically generates a ".tex" file (LaTex report), which can be externally processed to generate a PDF file.

ggplot objects for all figures are generated and saved in an R object ("module_output_plots.rda") within the output directory.

See Also

query_genomes_to_modules, analysis_pca_mean_distance_grouping,plot_heatmap

analysis_pca_mean_distance_calculation

Given a set of coordinates, calculate the mean distance between all points.

Description

Given a set of coordinates, calculate the mean distance between all points.

Usage

```
analysis_pca_mean_distance_calculation(MATRIX)
```

Arguments

MATRIX	- matrix with the rows refering to the points with N columns containing the coordinates (and therefore N dimensions).

Details

The mean distance of p points is calculated by the sum of the individual Euclidean distances divided by the total number of distances (given by $p * (p - 1)/2$).

Value

The mean distance (numeric).

Examples

```
data(data_example_moduleIDs)
data(data_example_genomeIDs)

# Calculate the module completion fraction (mcf) for the genomes and modules contained in the data objects above
OUT      <- query_genomes_to_modules(data_example_genomeIDs,MODULE_ID = data_example_moduleIDs)

pca <- prcomp(OUT$MATRIX)

mean_dist <- analysis_pca_mean_distance_calculation(pca$x)
# [1] 0.4805169
```

analysis_pca_mean_distance_grouping

Given a set of coordinates and a grouping variable, the mean distance is calculated for each group.

Description

Given a set of coordinates and a grouping variable, the mean distance is calculated for each group.

Usage

```
analysis_pca_mean_distance_grouping(MATRIX, FACTOR, factor_labs = NULL,
  nDim = 2, plot_mean_dist = T, Filename = "plot_mean_dist.pdf",
  plot_top_percent = NULL, ...)
```

Arguments

MATRIX - two column matrix with the point coordinates. Only the first two columns will be used.

FACTOR - used to split the data into groups. Character vector (or list of vetors) with the same length as rows in MATRIX.

factor_labs - optional. Character vector of the length of FACTOR if FACTOR is a list or length 1 if FACTOR is a character vector. Default (NULL; FACTOR names or letters used).

nDim - optional. Numeric vector of dimensions to use for the point coordinates. Default (2).

plot_mean_dist - logical. Should a plot of the mean distances be generated? Default (TRUE).

Filename - optional. Filename with path and extension. String added to distinguish FACTOR groups. Default ("plot_mean_dist.pdf" saved to working directory).

plot_top_percent - optional. Numeric vector between 0 and 1 or integer. Default (NULL, i.e. not plotted).

... - further arguments for plot_scatter().

Details

The mean distance of groups with one member (derived from the FACTOR labels) cannot be calculated.

nDim must be larger than 2.

plot_top_percent refers to the fraction or number of groups with the highest calculated mean distance that should be plotted on a scatter plot. Items are recycled if not as many as the number of grouping factors is provided.

Value

List containing a table of the mean distances (columns for group name, mean distance and size of group (N)) and a ggplot object of the data (NAs removed) if plot_mean_dist is TRUE.

See Also

analysis_pca_mean_distance_calculation

Examples

```
data(data_example_moduleIDs)
data(data_example_genomeIDs)

# Calculate the module completion fraction (mcf) for the genomes and modules contained in the data objects above
OUT      <- query_genomes_to_modules(data_example_genomeIDs,MODULE_ID = data_example_moduleIDs)

pca <- prcomp(OUT$MATRIX)
```

```
# Group data
this_FACTOR      <- rep(LETTERS[1:5],length(data_example_genomeIDs)/5)
mean_dist_output <- analysis_pca_mean_distance_grouping(pca$x,this_FACTOR,xLabs_angle = F,Width = 2, Height =
```

data_example_ECnumbers_vector

Example of a vector containing Enzyme Commission (EC) numbers.

Description

A dataset containing an example of a vector containing Enzyme Commission (EC) numbers that can be used with the function `misc_geneVector_module()` or by the function `query_genomes_to_modules()` after formatting it into a data frame (see function description).

Usage

```
data_example_ECnumbers_vector
```

Format

A character vector with 568 entries

ECs 1.1.1.10, 1.1.1.102, 1.1.1.105, 1.1.1.12, 1.1.1.14, 1.1.1.153, ...

Details

When generating an EC number vector, make sure that all entries have the 4 nomenclature positions (".-" denotes an unspecified field, e.g. "1.1.1.-").

Source

http://www.kegg.jp/kegg-bin/get_htext?eco00001 and expand the subsections to see the ECs.

See Also

data_example_KOnumbers_vector

Examples

```
# Load data
data("data_example_ECnumbers_vector")
head(data_example_ECnumbers_vector)
# [1] "1.1.1.10"  "1.1.1.102" "1.1.1.105" "1.1.1.12"  "1.1.1.14"  "1.1.1.153"
```

`data_example_genomeIDs` *Example of vector containing KEGG genome IDs to be used by* `query_genomes_to_modules()`.

Description

A dataset containing an example of a character vector that can be used as the `GENOME_INFO` input to the function `query_genomes_to_modules()`.

Usage

```
data_example_genomeIDs
```

Format

An object of class `character` of length 25.

See Also

plot_heatmap

Examples

```
# Load data
data("data_example_genomeIDs")
head(data_example_genomeIDs)
# [1] "T04503" "T04203" "T03253" "T00526" "T00341" "T00552"
```

`data_example_KOnumbers_vector` *Example of a vector containing KEGG Ortholog (KO) identifiers*

Description

A dataset containing an example of a vector containing KEGG Orthologs (K numbers or KOs) that can be used with the function `misc_geneVector_module()` or by the function `query_genomes_to_modules()` after formatting it into a data frame (see function description).

Usage

```
data_example_KOnumbers_vector
```

Format

A character vector with 2896 entries

KOs K00005, K00009, K00012, K00013, K00014, ...

Source

http://www.kegg.jp/kegg-bin/get_htext?eco00001 and expand the subsections to see the KOs.

See Also

data_example_ECnumbers_vector

Examples

```
# Load data
data("data_example_KOnumbers_vector")
head(data_example_KOnumbers_vector)
# [1] "K00005" "K00009" "K00012" "K00013" "K00014" "K00024"
```

---

data_example_moduleIDs

Example of vector containing KEGG module IDs to be used by `query_genomes_to_modules()`.

Description

A dataset containing an example of a character vector that can be used as the `MODULE_ID` input to the function `query_genomes_to_modules()`.

Usage

```
data_example_moduleIDs
```

Format

An object of class `character` of length 6.

See Also

plot_heatmap

Examples

```
# Load data
data("data_example_moduleIDs")
head(data_example_moduleIDs)
# [1] "M00356" "M00357" "M00563" "M00567" "M00596" "M00377"
```

data_example_multi_ECs_KOs

Example of a data frame with multiple datasets.

Description

A data frame containing an example of multiple datasets that could be analysed with the function misc_geneVector_module() or by the function query_genomes_to_modules() after formatting it into a data frame (see function description). The dataset entries are to illustrate what the KEGG genome data looks like.

Usage

```
data_example_multi_ECs_KOs
```

Format

A data frame with the following columns:

```
 ID       - KEGG genome ID, T number (e.g. "T00001")
 ORG_ID   - KEGG organism ID,3-4 letter code (e.g. "eco")
 ORGANISM - Organism name (e.g. "Escherichia coli K-12 MG1655")
KOs     - Concatenated string with the K numbers (e.g. "K00013;K00014;K00018;...").
 ECs       - Concatenated string with the EC numbers
             (e.g. "1.1.1.1;1.1.1.100;1.1.1.130;...")
```

Details

The columns 'KOs' or 'ECs' need to be pointed at to do the module mapping and is specified by using the argument 'mapBy' in query_genomes_to_modules() and The default is to use the column named 'KOs', but it can be named differently and specified in the argument field.

Specify the character to be used to split the string with the K/EC numbers using the argument 'split_vector_by' in query_genomes_to_modules()

See Also

query_genomes_to_modules

Examples

```
# Load data
data("data_example_multi_ECs_KOs")
head(data_example_multi_ECs_KOs)
```

data_example_sunburst *Example of the hierarchical data needed to generate a sunburst plot.*

Description

A dataset containing an example of a data frame that can be used with the function `plot_sunburst()`. This function helps visualize hierarchical information by making concentric "donut" plots which has the highest hierarchical level data at the centre going out.

Usage

```
data_example_sunburst
```

Format

An object of class `data.frame` with 8 rows and 4 columns.

See Also

plot_sunburst

Examples

```
# Load data
data("data_example_sunburst")
head(data_example_sunburst)
#        CLASS_I                  CLASS_II          CLASS_III                               NAME_SHORT
# 1 Pathway module Carbohydrate and lipid metabolism Fatty acid metabolism                            beta-Oxid
# 2 Pathway module            Energy metabolism   Methane metabolism                  Methanogenesis, from meth
# 3 Pathway module            Energy metabolism   Methane metabolism                   Methanogenesis, from acet
# 4 Pathway module            Energy metabolism     Carbon fixation Reductive acetyl-CoA pathway (Wood-Ljungdah
# 5 Pathway module            Energy metabolism  Nitrogen metabolism                 Dissimilatory nitrate redu
# 6 Pathway module            Energy metabolism   Methane metabolism                Methanogenesis, from methyl
```

data_example_sunburst_fill_by *Example of vector to provide alternative values to fill the outermost level of a sunburst plot.*

Description

A dataset containing an example of a numerical vector that can be used as the `fill_by` input to the function `plot_sunburst()`.

Usage

```
data_example_sunburst_fill_by
```

Format

An object of class `numeric` of length 8.

See Also

plot_sunburst

Examples

```
# Load data
data("data_example_sunburst_fill_by")
head(data_example_sunburst_fill_by)
# M00087 M00356 M00357 M00377 M00530 M00563
#    0.0    1.0    0.8    0.0    0.0    0.5
```

---

data_module_shortName_mapping

Data frame used to populate the short name in module_reference_table

---

Description

This data frame contains the following columns and is used to provide a manually abbreviated name to the modules to ease plotting.

Usage

```
data_module_shortName_mapping
```

Format

A data frame with the following columns:

```
ID         - KEGG module ID, M number (e.g. "M00001")
NAME       - KEGG module name
NAME_SHORT - Manually abbreviated KEGG module name (unique entries)
```

See Also

parseKEGG_module

Examples

```
# Load data
data("module_shortName_mapping")
head(module_shortName_mapping)
```

`misc_axisRound` *Find best value to round axis to.*

Description

Find best value to round axis to.

Usage

```
misc_axisRound(Vector, roundBy = NULL, Min = NULL)
```

Arguments

`Vector`	- values that are being plotted.
`roundBy`	- optional. The value to use for rounding. Default (`NULL`).
`Min`	- optional. The value to use as the starting value for the axis. Default (`NULL`).

Value

List containing the minimum and maximum axis values (`$min` and `$max`, respectively) and the array that can be used to indicate the axis breaks (`$array`)

`misc_check_duplicate_names` *Check for duplicated strings and append letter code to distinguish them.*

Description

Check for duplicated strings and append letter code to distinguish them.

Usage

```
misc_check_duplicate_names(NAME_VECTOR)
```

Arguments

`NAME_VECTOR`	- character vector.

`misc_create_labels` *Creates labels to add to a character vector to make all entries unique*

Description

Creates a label vector of length N, such that they are unique and help distinguish them

Usage

```
misc_create_labels(N)
```

Arguments

`N`	- numeric. Indicates the length of the label vector to be return.

Value

A character vector of length N with unique set of alpha-numeric labels.

`misc_evaluate_block` *Evaluate a KEGG module block definition.*

Description

Evaluate a KEGG module block definition given a vector of genes (KEGG Orthologs -KOs-, identified with K number). Some KOs can be mapped to Enzyme Commission (EC) numbers.

Usage

```
misc_evaluate_block(gene_vector, BLOCK, KO_in_DEF_EC = FALSE)
```

Arguments

`gene_vector`	- vector containing either K or EC numbers. See Details.
`BLOCK`	- KEGG module DEFINITION BLOCK. See parseKEGG_module.
`KO_in_DEF_EC`	- logical. If enzyme IDs are given, should lingering K numbers in the module DEFINITION be assumed to be present? Default (`FALSE`).

Details

`gene_vector` must contain either K or EC numbers.

misc_geneVector_module

Map a list of EC or K numbers to KEGG modules.

Description

This function maps the list of Enzyme Commission (EC) numbers or KEGG orthologs (specified as K numbers) given by gene_vector to the KEGG modules using an in-build reference table (module_reference_table) of all the KEGG modules in the database at the time of release.

Usage

```
misc_geneVector_module(gene_vector, MODULE_ID, SEARCH_NAME, SEARCH_CLASS_I,
  SEARCH_CLASS_II, SEARCH_CLASS_III, EXCLUDE_NAME,
  use_module_reference_table = NULL, ...)
```

Arguments

gene_vector	- character vector listing EC or K numbers. If more K numbers are given, the K number-based definition will be used. See Value.
MODULE_ID	- optional. Character vector listing specific module IDs (e.g. M00001).
SEARCH_NAME	- optional. Character vector listing terms to search in NAME field (case-insensitive).
SEARCH_CLASS_I	- optional. Character vector listing terms to search in CLASS_I field (case-insensitive).
SEARCH_CLASS_II	- optional. Character vector listing terms to search in CLASS_II field (case-insensitive).
SEARCH_CLASS_III	- optional. Character vector listing terms to search in CLASS_III field (case-insensitive).
EXCLUDE_NAME	- optional. Character vector listing terms that if matched in NAME field will be excluded (case-insensitive).
use_module_reference_table	- optional. Provide a data frame with updated KEGG module database OR with custom-made modules. Default (NULL; inbuilt data used). See Details.
...	- further arguments such as KO_in_DEF_EC. See Details.

Details

The modules to be analysed are determined by search queries. These can search across the module NAME, CLASS (I-III) or by specifying the module ID (M number). See the Argument section. When none of the optional arguments are specified, the default is to search all modules. An additional argument, EXCLUDE_NAME, can be used to exclude modules with a certain term in the NAME field. For example, specifying EXCLUDE_NAME = "biosynthesis" would return the search across all modules, except those that contain "biosynthesis" in the name. For instance, module "M00005" named "PRPP biosynthesis, ribose 5P => PRPP" and 168 others would be excluded form the search.

If neither MODULE_ID or at least one SEARCH term is specified, the default is to analyse all modules.

The use_ argument allow users with KEGG FTP access to provide the updated data from the KEGG databases in the form of reference tables AND/OR for advanced users to provide custom-made modules (see below). These reference tables can be generated with `parseKEGG_module` and need to have a specific format (see function descriptions for details on format).

The module definition (contained in `module_reference_table`) describes the relationship between genes and modules and is used to identify the modules in which gene is involved. The user can provide custom-made module definitions that use the logical expression format (however, the table format must be conserved!).

Value

List: `$FRACTION` matrix containing the completeness fraction for the modules matched in the query along the columns.

`$METADATA` data frame with the metadata for the modules matched in the query. Columns:

```
    (1)     MODULE_ID          - M number (e.g. M00001);
    (2)     MODULE_NAME        - the KEGG given name for the module;
    (3)     MODULE_NAME_SHORT  - manually shortened module name (for plotting
                                 purposes);
(4 - 6)     CLASS_I - III      - hierarchical module classes;
   (7)     DEFINITION        - KEGG module definition in terms of K or EC numbers
                                 (without optional K or EC numbers);
   (8)     OPTIONAL          - the optional K or EC numbers that are part of the
                                 KEGG module definition (NA otherwise);
   (9)    KOs_IN_DEF        - logical flag indicating whether there are K numbers
                              involved in that module definition (if EC numbers
                                 are given);
```

`$ADD_INFO` data frame with additional output information. Columns:

```
(1) MODULE_ID            - M number (e.g. M00001);
(2) MODULE_NAME_SHORT   - manually shortened module name (for plotting purposes);
(3) FRACTION           - the number of complete blocks divided by the total number
                           of blocks;
(4) nBLOCKS              - the number of blocks that make up the KEGG module;
(6) COVERAGE           - the K numbers or ECs that are present and involved in the
                           KEGG module definition;
(7) OPTIONAL_PRESENT    - the K numbers or ECs that are present and are listed as
                           optional;
```

See parseKEGG_module or visit http://www.genome.jp/kegg/module.html for more information.

See Also

parseKEGG_module

Examples

```
output_module_ECs <- misc_geneVector_module(data_exampleECnumbers_vector,
                                            MODULE_ID=paste("M0000",1:5,sep=""))

output_module_KOs <- misc_geneVector_module(data_exampleKOnumbers_vector,
                                            MODULE_ID=paste("M0000",1:5,sep=""))
```

`misc_module_definition_block_EC` *KEGG module definition processing: subblocks*

Description

Format the KEGG module database - Format the KEGG module DEFINITION for ease of analysis by excluding optional KEGG orthologs. Used by `parseKEGG_module()`.

Usage

```
misc_module_definition_block_EC(BLOCK, ORTHOLOGS)
```

Arguments

`BLOCK`	- string containing a block of a KEGG module DEFINITION (logical expression using K numbers).
`ORTHOLOGS`	- vector listing the K numbers and related EC numbers.

Details

The KEGG module definition uses optional KEGG orthologs, indicated by a "-". These are removed and the corresponding EC number stored.

See Also

parseKEGG_module

`misc_module_definition_check` *KEGG module definition processing: block formatting.*

Description

Format the KEGG module DEFINITION for ease of analysis.

Usage

```
misc_module_definition_check(DEFINITION)
```

Arguments

`DEFINITION`	- string containing a KEGG module DEFINITION (logical expression using K numbers).

Details

The KEGG module definition uses both spaces and plus signs to indicate `'AND'` operations. However, the `'AND'` operation can be used to split the module definition into BLOCKS or to indicate molecular complex composition. To simplify analysis, we will use spaces ONLY to delimit KEGG module BLOCKS and the plus sign ONLY to indicate molecular complexes and `'AND'` operations within blocks.

See Also

parseKEGG_module

`misc_module_definition_optional` *KEGG module definition processing: optional KEGG ortholog exclusion*

Description

Format the KEGG module database - Format the KEGG module definition for ease of analysis by excluding optional KEGG orthologs. Used by `parseKEGG_module()`.

Usage

```
misc_module_definition_optional(BLOCK, ORTHOLOGS)
```

Arguments

`BLOCK`	- string containing a block of a KEGG module DEFINITION (logical expression using K numbers).
`ORTHOLOGS`	- vector listing the K numbers and related EC numbers.

Details

The KEGG module definition uses optional KEGG orthologs, indicated by a "-". These are removed and the corresponding EC number stored.

Value

A list with the formatted block (`$thisBlock`) and any optional K and EC numbers (`$thisOptional_KO` and `$thisOptional_EC`)

See Also

parseKEGG_module

`misc_module_subgroup_indexing` *Subgroup check - formatting KEGG module definition.*

Description

This functions helps format the KEGG module DEFINITION by checking all the bracket-delimitted BLOCKS to then remove flanking brackets (unnecessary).

Usage

```
misc_module_subgroup_indexing(DEFINITION)
```

Arguments

`DEFINITION`	- KEGG module definition

See Also

parseKEGG_module,misc_module_definition_check

`parseKEGG_compound` *Parse the KEGG compound database*

Description

Read the KEGG compound database text file and format it into a reference table.

Usage

```
parseKEGG_compound(KEGG_path, outDir = "output", verbose = T, ...)
```

Arguments

`KEGG_path`	- string pointing to the location of the KEGG database parent folder. The path to the required file is contained within the function.
`outDir`	- string pointing to the output folder. Default ("output/").
`verbose`	- logical. Should progress be printed to the screen? Default (`TRUE`).
`...`	- other arguments for `parseKEGG_file()`.

Details

The columns are automatically generated by the `parseKEGG_file` function into variables, which are further formatted specifically for the KEGG compound database.

The text file used is "`KEGG_path`/ligand/compound/compound".

It decompresses "`KEGG_path`/ligand/compound.tar.gz" if needed.

Value

Generates compound_reference_table (.txt & .rda; saved to `'outDir'`) and returns a data frame with as many rows as entries and the following columns (or variables):

```
 (1) ID         - C number identifier (e.g. "C00001");
 (2) NAME       - compund name(s);
 (3) FORMULA    - chemical formula;
 (4) EXACT_MASS - compound's mass;
 (5) MOL_WEIGHT - molecular weight;
 (6) REMARK     - relationship with D number and others;
 (7) REACTION   - reactions IDs (R#####) in which the compound is involved;
 (8) PATHWAY   - pathway(s) in which the compound is involved (map### and name);
 (9) MODULE    - module(s)  in which the compound is involved (M##### and name);
(10) ENZYME   - EC numbers catalysing a reaction in which the compound is involved;
 (11) BRITE;    (12) DBLINKS;   (13) ATOM;      (14) BOND;  (15) COMMENT;
 (16) BRACKET;  (17) SEQUENCE;  (18) REFERENCE;
```

In all instances, multiple entries in a given column are separated by '[;]'. EC numbers are of the form `'\d[.]\d+[.]\d+[.]\d+'` (e.g. '1.97.1.12')

See Also

parseKEGG_file

Examples

```
KEGG_path <- "~/KEGG" # MODIFY TO KEGG PARENT FOLDER!
# The parent folder should contain the following (KEGG FTP structure):
# brite/
# genes/
# ligand/
# medicus/
# module/
# pathway/
# README.kegg
# RELEASE
# xml/

compound_reference_table <- parseKEGG_compound(KEGG_path)
 # A .txt file (tab separated) is written to output/ (relative to current working directory)
```

`parseKEGG_enzyme` *Parse the KEGG enzyme database*

Description

Read the KEGG enzyme database text file and format it into a reference table.

Usage

```
parseKEGG_enzyme(KEGG_path, outDir = "output", verbose = T, ...)
```

Arguments

`KEGG_path` - string pointing to the location of the KEGG database parent folder. The path to the required file is contained within the function.

`outDir` - string pointing to the output folder. Default ("output/").

`verbose` - logical. Should progress be printed to the screen? Default (`TRUE`)

`...` - other arguments for `parseKEGG_file()`.

Details

The columns are automatically generated by the `parseKEGG_file` function into variables, which are further formatted specifically for the KEGG enzyme database.

The text file used is "`KEGG_path`/ligand/enzyme/enzyme".

It decompresses "`KEGG_path`/ligand/enzyme.tar.gz" if needed.

Value

Generates enzyme_reference_table (.txt & .rda; saved to `'outDir'`) and returns a data frame with as many rows as entries and the following columns (or variables):

```
(1) ID         - Enzyme Commission (EC) number (e.g. "1.1.1.1"; 4 positions);
(2) NAME       - enzyme name(s);
(3) CLASS_I    - enzyme class; refers to the first position.
                  CLASSES:
                   1. Oxidoreductases, 2. Transferases, 3. Hydrolases,
                   4. Lyases,          5. Isomerases,   6. Ligases
(4) CLASS_II   - further enzyme class info; refers to the second position
                  (different for every class);
(5) CLASS_III  - further enzyme class info; refers to the third  position
                  (different for every class);
(6) SYSNAME    - alternative names;
(7) REACTION   - reaction(s) the enzyme catalyses;
(8) ALL_REAC   - reaction ID(s) (R number);
(9) SUBSTRATE  - substrate name and ID(s);
(10) PRODUCT   - product   name and ID(s);
(11) COMMENT;     (12) HISTORY;   (13) REFERENCE;
(14) PATHWAY   - pathway(s) in which the enzyme is involved (ec### and name);
(15) ORTHOLOGY - related KEGG Ortholog(s) (K number; K##### and name);
(16) GENES;       (17) DBLINKS;

*In all instances, multiple entries in a given column are separated by '[;]'.
```

See Also

parseKEGG_file

Examples

```
KEGG_path <- "~/KEGG" # MODIFY TO KEGG PARENT FOLDER!
# The parent folder should contain the following (KEGG FTP structure):
# brite/
# genes/
# ligand/
# medicus/
# module/
# pathway/
# README.kegg
# RELEASE
# xml/

enzyme_reference_table <- parseKEGG_enzyme(KEGG_path)
 # A .txt file (tab separated) is written to output/ (relative to current working directory)
```

parseKEGG_execute_all *Execute all parseKEGG parent functions to format KEGG databases into data frames*

Description

Execute all parseKEGG parent functions to format specific KEGG databases into data frames.

Usage

```
parseKEGG_execute_all(KEGG_path, ...)
```

Arguments

KEGG_path	- string pointing to the location of the KEGG database parent folder.
...	- other arguments, such as outDir, for parseKEGG_file, parseKEGG_file.list and database-specific functions (below).

See Also

parseKEGG_compound, parseKEGG_enzyme, parseKEGG_genome, parseKEGG_module,

parseKEGG_ko, parseKEGG_reaction, parseKEGG_ko_enzyme, parseKEGG_ko_reaction

Examples

```
KEGG_path <- "~/KEGG" # MODIFY!
parseKEGG_parseKEGG_execute_all(KEGG_path)
# multiple reference_table objects in workspace and .txt files written to output/ (relative to current working c
```

parseKEGG_file *Parse any KEGG file without extension*

Description

Read the KEGG database text file without extension (e.g. 'module', 'enzyme', 'genome') and format it into a reference table. Generates DATABASE_reference_table (data frame).

Usage

```
parseKEGG_file(FILE_PATH, split_pattern = "ENTRY", pathway_trim = T,
  verbose = T)
```

Arguments

split_pattern	- string to use to identify start of new section/entry. Default ("ENTRY").
pathway_trim	- logical. Should the file 'KEGG_path/pathway/pathway' be trimmed to only include 'map' entries? (i.e. exclude ko, ec and organism-specific pathways). Default (TRUE).
verbose	- logical. Should progress be printed to the screen? Default (TRUE).
FILE	- string pointing to the location of the file WITHOUT EXTENSION (uncompressed).

Details

File must be decompressed before being processed. The functions that use this function (listed below in see also) perform this step if necessary.

Value

A data frame with the formatted data.See the file-specific functions.

See Also

parseKEGG_compound,parseKEGG_enzyme,parseKEGG_genome, parseKEGG_module, parseKEGG_reaction, parseKEGG_execute_all

Examples

```
compound_file_path <- "~/KEGG/ligand/compound/compound" # MODIFY!
reference_table <- parseKEGG_file(compound_file_path)
# WITHOUT DATABASE SPECIFIC FORMATTTING!
```

`parseKEGG_file.list` *Parse any '.list' KEGG file*

Description

Reads the KEGG database text files with '.list' extension (e.g. 'ko_enzyme.list', 'ko_reaction.list') and formats it into a matrix with a binary indicator or relationships or mappings.

Usage

```
parseKEGG_file.list(FILE_PATH)
```

Arguments

`FILE_PATH`	- string pointing to the location of '.list' file.

Value

MATRIX

See Also

parseKEGG_ko_enzyme, parseKEGG_ko_reaction

Examples

```
ko_enzyme_file_path <- "~/KEGG/genes/ko/ko_enzyme.list" # MODIFY!
MAP <- parseKEGG_file(ko_enzyme_file_path)
```

`parseKEGG_genome` *Parse the KEEG genome database*

Description

Read and format the KEGG genome database text file and the organism information and format it into a reference table.

Usage

```
parseKEGG_genome(KEGG_path, outDir = "output", addKOs = T, addECs = T,
  includeViruses = F, verbose = T, ...)
```

Arguments

`KEGG_path` - string pointing to the location of the KEGG database parent folder. The path to the required file(s) is contained within the function.

`outDir` - optional. String pointing to the output folder. Default ("output/").

`addKOs` - logical. Should the list of K numbers be retrieved for each organism? Default (`TRUE`).

`addECs` - logical. Should the list of ECs be retrieved for each organism? Default (`TRUE`).

`includeViruses` - logical. Should viral genomes be included? Default (`FALSE`).

`verbose` - logical. Should progress be printed to the screen? Default (`TRUE`).
@param ... - other arguments for `parseKEGG_file()`.

Details

The columns are automatically generated by the `parseKEGG_file` function into variables, which are further formatted specifically for the KEGG genome database.

The main text file used is "`KEGG_path`/genes/genome/genome".

It decompresses "`KEGG_path`/genes/genome.tar.gz" if needed.

If `addECs` and/or `addKOs` are set to `TRUE`, the KEGG genome 3-4 letter code identifier are used to retrieve the enzyme and KEGG ortholog content for each KEGG genome, respectively.

Value

Generates genome_reference_table (.txt & .rda; saved to `'outDir'`) and returns a data frame with as many rows as entries and the following columns (or variables):

```
 (1) ID                     - KEGG genome   identifier (T0 number; e.g. "T00001");
(2) ORG_ID              - KEGG organism identifier (3 or 4 letter code; e.g. "hin");
 (3) STATUS                 - genome sequence status (e.g. "Complete Genome");
 (4) NAME                   - various identifiers (e.g. "hin, HAEIN, 71421");
 (5) ORGANISM               - organism name (Genus species sp);
(6) ANNOTATION            - type of annotation (one of "manual", "KOALA" or  "none");
 (7) TAXONOMY;               (8)  DATA_SOURCE;
 (9) ORIGINAL_DB            - original database;
 (10) KEYWORDS;              (11) DISEASE;
 (12) COMMENT;               (13) CHROMOSOME;
 (14) STATS_N_NUCLEOTIDES - statistics, number of nucleotides;
 (15) STATS_N_GENES_PROT  - statistics, number of protein-encoding genes;
 (16) STATS_N_GENES_RNA   - statistics, number of RNA-encoding genes;
 (17) REFERENCE          - reference(s) for study from which genomic sequence was
                              derived;
 (18) PLASMID;               (19) DBLINKS;
 (20) KOs                   - concatenated string of KEGG Orthologs (K numbers;
                              e.g. "K00001");
(21) ECs                - concatenated string of Enzyme Classification (EC) numbers
                              (e.g. "1.1.1.1").

    *In all instances, multiple entries in a given column are separated by '[;]'.
```

See Also

parseKEGG_file

Examples

```
KEGG_path <- "~/KEGG" # MODIFY TO KEGG PARENT FOLDER!
# The parent folder should contain the following (KEGG FTP structure):
# brite/
# genes/
# ligand/
# medicus/
# module/
# pathway/
# README.kegg
# RELEASE
# xml/

genome_reference_table <- parseKEGG_genome(KEGG_path)
 # A .txt file (tab separated) is written to output/ (relative to current working directory)
```

parseKEGG_ko *Parse the KEGG orthology (KO) database*

Description

Read and format the KEGG orthology (KO) database (containing ortholog (gene) information) text file into a reference table.

Usage

```
parseKEGG_ko(KEGG_path, outDir = "output", verbose = T, ...)
```

Arguments

KEGG_path	- string pointing to the location of the KEGG database parent folder. The path to the required file is contained within the function.
outDir	- optional. String pointing to the output folder. Default ("output/").
verbose	- logical. Should progress be printed to the screen? Default (TRUE).
...	- other arguments for parseKEGG_file().

Details

The columns are automatically generated by the parseKEGG_file function into variables, which are further formatted specifically for the KEGG ortholog database.

The text file used is "KEGG_path/genes/ko/ko".

It decompresses "KEGG_path/genes/ko.tar.gz" if needed.

Value

Generates ko_reference_table (.txt & .rda; saved to 'outDir') and returns a data frame with as many rows as entries and the following columns (or variables):

```
*In all instances, multiple entries in a given column are separated by '[;]'.
```

See Also

parseKEGG_file

Examples

```
KEGG_path <- "~/KEGG" # MODIFY TO KEGG PARENT FOLDER!
# The parent folder should contain the following (KEGG FTP structure):
# brite/
# genes/
# ligand/
# medicus/
# module/
# pathway/
# README.kegg
# RELEASE
# xml/

ko_reference_table <- parseKEGG_ko(KEGG_path)
 # A .txt file (tab separated) is written to output/ (relative to current working directory)
```

parseKEGG_ko_enzyme *Map KEGG orthology (KO) to Enzyme Commission (EC) numbers*

Description

Map KEGG orthologs (K numbers) to EC numbers and format it into a matrix with binary indicator for mapping/relationship. Generates ko_enzyme_map (.txt & .rda)

Usage

```
parseKEGG_ko_enzyme(KEGG_path, outDir = "output", verbose = T)
```

Arguments

KEGG_path	- string pointing to the location of the KEGG database parent folder. The path to the required file is contained within the function.
outDir	- string pointing to the output folder. Default ("output/").
verbose	- logical. Should progress be printed to the screen? Default (TRUE)

Value

Data frame establishing the relationship between K numbers and enzymes (binary).

```
> ko_enzyme_map[1:3,1:3]

           1.1.1.1 1.1.1.10 1.1.1.100
K00001        1        0         0
K00002       0        0         0
K00003       0        0         0
```

See Also

parseKEGG_file.list

Examples

```
KEGG_path     <- "~/KEGG" # MODIFY TO KEGG PARENT FOLDER!
# The parent folder should contain the following (KEGG FTP structure):
# brite/
# genes/
# ligand/
# medicus/
# module/
# pathway/
# README.kegg
# RELEASE
# xml/

ko_enzyme_map <- parseKEGG_ko_enzyme(KEGG_path)
 # A .txt file (tab separated) is written to output/ (relative to current working directory)
```

parseKEGG_ko_reaction *Map KEGG orthologs (KOs) to Reaction IDs*

Description

Map KEGG orthologs (KOs) to Reaction IDs and format it into a matrix with binary indicator for mapping/relationship. Generates 'ko_reaction_map' (.txt & .rda).

Usage

```
parseKEGG_ko_reaction(KEGG_path, outDir = "output", verbose = T)
```

Arguments

KEGG_path	- string pointing to the location of the KEGG database parent folder. The path to the required file is contained within the function.
outDir	- string pointing to the output folder. Default ("output/").
verbose	- logical. Should progress be printed to the screen? Default (TRUE).

Value

Data frame establishing the relationship between KO numbers and reactions (R number) (binary).

```
> ko_reaction_map[1:3,1:3]

              R00005 R00006 R00008
    K00001      0      0      0
    K00002      0      0      0
    K00003      0      0      0
```

See Also

parseKEGG_file.list

Examples

```
KEGG_path     <- "~/KEGG" # MODIFY TO KEGG PARENT FOLDER!
# The parent folder should contain the following (KEGG FTP structure):
# brite/
# genes/
# ligand/
# medicus/
# module/
# pathway/
# README.kegg
# RELEASE
# xml/

ko_reaction_map <- parseKEGG_ko_reaction(KEGG_path)
 # A .txt file (tab separated) is written to output/ (relative to current working directory)
```

parseKEGG_module *Parse the KEGG module database*

Description

Read the KEGG module database text file and format it into a reference table.

Usage

```
parseKEGG_module(KEGG_path, outDir = "output", verbose = T,
  shortName_file_path = "", ...)
```

Arguments

`KEGG_path` - string pointing to the location of the KEGG database parent folder. The path to the required file is contained within the function.

`outDir` - string pointing to the output folder. Default ("output/"). `NULL` overwrites existing files.

`verbose` - logical. Should progress be printed to the screen? Default (`TRUE`)

`shortName_file_path` - file path to table containing Module IDs (column 1), Short name (column 2). Default ("")

`...` - other arguments for `parseKEGG_file()`

Details

The columns are automatically generated by the `parseKEGG_file` function into variables, which are further formatted specifically for the KEGG module database.

The text file used is "`KEGG_path`/module/module".

It decompresses "`KEGG_path`/module/module.gz" if needed.

DEFINITION: Logical Expression (adapted from http://www.genome.jp/kegg/module.html)

The MODULEs (identified by an M number; e.g. "M00001") are defined by a logical expression (DEFINITION) of KEGG orthologs (KO numbers, KOs; e.g. "K00001") and sometimes other M numbers, facilitating the automatic evaluation of whether a module is complete in a given genome.

A MODULE is made up of BLOCKS (SPACE delimited). Each block is defined by a logical expression to determine which KOs are needed in the definition. All BLOCKS must be present to be able to state that a MODULE is COMPLETE.

The KEGG module DEFINITION has been formatted to simplify its use, but the logical expression is conserved. Where space or a plus sign represent AND operations in the KEGG definition, we have replaced all instances with `'&'`. Similarly, we have replaced all comas (used within KEGG to represent an OR operation) with pipes (`'|'`).

We have also translated the K number based DEFINITION to an enzyme based DEFINITION using the ORTHOLOGY information. K numbers can be mapped to enzymes using the Enzyme Classification (EC) numbers (redundancy expected). Note that not all K numbers have an association to an EC number. In these instances, the K numbers have been left in the MODULE DEFINITION. When evaluating whether a MODULE is complete using EC numbers, the user must decide whether to assume that those genes are present or not (deafult). See query_genomes_to_modules

Optional items in the complex or definition (denoted by a minus sign) have been removed from the main definition and listed as a separate column under `'OPTIONAL_' (EC or K numbers)`.

`shortName_file_path` can be used to provide the path to a file listing:

```
Column named 'ID'         - KEGG module ID (M numbers),
Column named 'NAME'       - KEGG module NAME,
Column names 'NAME_SHORT' - manually abbreviated names for plotting purposes.
```

`NAME_SHORT` abbreviations: 2-CRS, two-component regulatory system; PS, photosystem/photosynthesis; pwy, pathway; TS, transport system; R, resistance;

Value

Generates module_reference_table (.txt & .rda; saved to `'outDir'`) and returns a data frame with as many rows as entries and the following columns (or variables):

```
     (1) ID             - KEGG module identifier (M number; e.g. "M00001");
     (2) NAME           - KEGG module name;
    (3) NAME_SHORT     - abbreviated module name (for visualization purposes);
   (4) DEFINITION_KOs - module definition as a logical expression (see Details)
                          in terms of KEGG Orthologs;
   (5) DEFINITION_ECs - module definition as a logical expression (see Details)
                          in terms of EC numbers;
     (6) OPTIONAL_KOs   - optional K numbers listed;
     (7) OPTIONAL_ECs   - optional EC numbers listed;
     (8) ORTHOLOGY      - relationship between K and EC numbers;
 (9 - 11) CLASS_I - III  - hierarchical module classes;
```

```
        (12) PATHWAY       - pathway(s)  in which the module is involved (map### and
                                name);
        (13) REACTION      - reaction(s) in which the module is involved (R##### and
                                their corresponding compound IDs);
         (14) COMPOUND       - compund(s)  in which the module is involved (C#####);
         (15) DBLINKS;         (16) RMODULE;         (17) REFERENCE;
         (18) COMMENT;         (19) BRITE;

     *In all instances, multiple entries in a given column are separated by '[;]'.
```

See Also

parseKEGG_file, misc_module_definition_check,

misc_module_definition_optional,' misc_module_definition_block_EC

Examples

```
KEGG_path <- "~/KEGG" # MODIFY TO KEGG PARENT FOLDER!
# The parent folder should contain the following (KEGG FTP structure):
# brite/
# genes/
# ligand/
# medicus/
# module/
# pathway/
# README.kegg
# RELEASE
# xml/

module_reference_table <- parseKEGG_module(KEGG_path)
 # And .txt file written to output/ (relative to current working directory)
```

parseKEGG_process_KEGG_taxonomy

Retrieve the taxonomy listed as part of the KEGG genome database.

Description

Retrieve the taxonomy listed as part of the KEGG genome database.

Usage

```
parseKEGG_process_KEGG_taxonomy(genome_reference_table,
  taxonomy_header = "TAXONOMY", org_header = 1)
```

Arguments

`genome_reference_table`
: - table containing the different genome entries (rows) and data (columns). See parseKEGG_genome.

`taxonomy_header`
: - optional. Character with header name or number indicating column index for taxonomic information. Default ("TAXONOMY").

`org_header`
: - optional. Character with header name or number indicating column index for organism name or ID. Use to name the output rownames. Default (1; first column).

Details

After processing the KEGG genome database, this function can extract the taxonomic information. This is obtained from splitting the `TAXONOMY` column after the "LINEAGE" tag into "words". This information should specify KINGDOM, PHYLUM, CLASS, ORDER, FAMILY, GENUS (6), but there are ocasionally more or less words than expected. Therefore NOTE that it is incomplete and inconsistent, most likely because it is derived from multiple sources. Use with caution.

Value

Data frame with 6 columns containing the taxonomic information (columns: KINGDOM, PHYLUM, CLASS, ORDER, FAMILY, GENUS). The rownames contain the organism information (name or ID) as specified with `org_header`. `UNKNOWN` label is added where no information is available.

See Also

parseKEGG_genome

Examples

```
# Generate KEGG's genome database table
genome_reference_table <- parseKEGG_genome("~/KEGG")

# Extract taxonomic information from genome_referene_table
TAXONOMY_table <- parseKEGG_process_KEGG_taxonomy(genome_reference_table)
head(TAXONOMY_table)
#        KINGDOM         PHYLUM             CLASS          ORDER            FAMILY           GENUS
# T00001  Bacteria Proteobacteria Gammaproteobacteria  Pasteurellales     Pasteurellaceae      Haemophilus
# T00002  Bacteria   Tenericutes         Mollicutes Mycoplasmataceae         Mycoplasma          UNKNOWN
# T00003   Archaea  Euryarchaeota      Methanococci  Methanococcales Methanocaldococcaceae Methanocaldococcus
# T00004  Bacteria  Cyanobacteria    Synechococcales Merismopediaceae      Synechocystis          UNKNOWN
# T00005 Eukaryota          Fungi          Dikarya      Ascomycota    Saccharomycotina  Saccharomycetes
# T00006  Bacteria   Tenericutes         Mollicutes Mycoplasmataceae         Mycoplasma          UNKNOWN
```

`parseKEGG_reaction` *Parse KEGG reaction database*

Description

Read and format the KEGG reaction database text file into a reference table.

Usage

```
parseKEGG_reaction(KEGG_path, outDir = "output", verbose = T, ...)
```

Arguments

KEGG_path	- string pointing to the location of the KEGG database parent folder. The path to the required file is contained within the function.
outDir	- string pointing to the output folder. Default ("output/").
verbose	- logical. Should progress be printed to the screen? Default (TRUE)
...	- other arguments for parseKEGG_file().

Details

The columns are automatically generated by the parseKEGG_file function into variables, which are further formatted specifically for the KEGG reaction database.

The text file used is "KEGG_path/ligand/reaction/reaction".

It decompresses "KEGG_path/ligand/reaction.tar.gz" if needed.

Value

Generates reaction_reference_table (.txt & .rda; saved to 'outDir') and returns a data frame with as many rows as entries and the following columns (or variables):

```
 (1) ID         - R number identifier (e.g. "R00001");
 (2) NAME       - reaction or enzyme name;
 (3) DEFINITION - reaction definition using compound's names;
 (4) EQUATION   - reaction definition using compound's IDs;
 (5) ENZYME     - Enzyme Commission (EC) number (e.g. "1.1.1.1");
 (6) COMMENT;  (7) RCLASS;
```

In all instances, multiple entries in a given column are separated by '[;]'.

See Also

parseKEGG_file

Examples

```
KEGG_path <- "~/KEGG" # MODIFY TO KEGG PARENT FOLDER!
# The parent folder should contain the following (KEGG FTP structure):
# brite/
# genes/
# ligand/
# medicus/
# module/
# pathway/
# README.kegg
# RELEASE
# xml/

reaction_reference_table <- parseKEGG_reaction(KEGG_path)
 # A .txt file (tab separated) is written to output/ (relative to current working directory)
```

plot_heatmap *Plot a heatmap*

Description

Plot a heatmap of the the module fraction present across genomes or of the variance across groups.

Usage

```
plot_heatmap(data_in, row_lab = "Genomes", col_lab = "Modules",
  ORDER_MATRIX = FALSE, legend_name = "Module\ncompleteness\nfraction\n",
  set_yLim = FALSE, Filename = "", Width = 24, Height = 18)
```

Arguments

`data_in`	- numeric matrix OR 3-column data frame. See `Details`.
`row_lab`	- optional. String to specify the axis label corresponding to the row values. Default ("Genomes").
`col_lab`	- optional. String to specify the axis label corresponding to the column values. Default ("Modules").
`ORDER_MATRIX`	- logical. Should rows and columns be reorder according to the dendogram (hierarchical clustering). Default (`FALSE`).
`legend_name`	- optional. String to specify the legend title. Default ("Fraction\n matched\n").
`set_yLim`	- optional. Logical or numeric (length 2) indicating whether to set the y limit of the heatmap scale. Default (`FALSE`). See Details.
`Filename`	- optional. A character vector containing the file path to save image(s). The device to save is determined by file extension. Default ("", i.e. file not written).
`Width`	- optional. Width size for file. Default (24 in).
`Height`	- optional. Height size for file. Default (18 in).

Details

If `data_in` is a data frame, the heatmap will be made using the first column as the row entry labels, the second column as the column entry labels and the third as the actual value to be plotted.

If `data_in` is a numeric matrix, it will be 'melted' into this type of data frame by using the `reshape2::melt` function.

`set_yLim` - #' the default (`FALSE`), sets the y limit to `c(0, 1)`, while `TRUE` to the `c(min, max)` of the data. If numeric, the values given are used for the lower and upper limits respectively.

Value

ggplot object. Saves figures if Filename is specified.

Examples

```
data(data_example_moduleIDs)
data(data_example_genomeIDs)

# Calculate the module completion fraction (mcf) for the genomes and modules contained in the data objects above
OUT      <- query_genomes_to_modules(data_example_genomeIDs,MODULE_ID = data_example_moduleIDs)

# Make a heatmap of the mcf output from query_genomes_to_modules
p <- plot_heatmap(OUT$MATRIX,Filename = ”../GitHub/MetQy/fig/plot_heatmap.png”)

# Reorder rows and columns according to the dendrogram resulting from hierarchical clustering (same data).
p <- plot_heatmap(OUT$MATRIX,ORDER_MATRIX = T,Filename = ”../GitHub/MetQy/fig/plot_heatmap_ordered.png”)
```

`plot_scatter` *Scatter plot*

Description

Scatter plot for pairs of categorical data with a numeric value.

Usage

```
plot_scatter(plot_data, X = 1, Y = 3, colBy = NULL, xLab = "",
  yLab = "", xLabs_angle = T, Filename = "plot_scatter.pdf", Width = 24,
  Height = 18)
```

Arguments

`plot_data` - data frame. The columns to be plotted are indicated with `X` and `Y`.Order given is conserved.

`X, Y` - optional. Number indicating column to use for x axis and y axis, respectively. Default (1 and 3).

`colBy` - optional. Number indicating column to use for coloring grouping. Default (`NULL`).

`xLab, yLab` - optional. String to use for x label and y label, respectively. Default (first and second column names, respectively).

`xLabs_angle` - optional. Should x-axis labels be rotated 45 degrees? Default (`TRUE`).

`Filename` - optional. String(s) to use for file name. Default ("plot_scatter.pdf"). If set to `""` a file is not written.

`Width` - width for figure file. Default (24in).

`Height` - height for figure file. Default (18in).

Details

This function is used by analysis_genomes_module_output() to plot the module variance accross genomes and

by analysis_pca_mean_distance_grouping() to plot the mean PCA distance of the groups analysed.

If `colBy` is set to `NULL`, no grouping will be done for colouring (a scale warning will be issued that can be safely ignored).

Value

ggplot2 plot object

See Also

analysis_genomes_module_output,analysis_pca_mean_distance_grouping

Examples

```
plot_data_example <- data.frame("Groups"=LETTERS[1:12],
                                "Factor"=c(rep(1,4),rep(2,4),rep(3,4)),
                                "Value"=runif(12,-10,10),stringsAsFactors = FALSE)

plot_scatter(plot_data_example,Filename = "")

# Change plot order to be according to the numeric value
plot_data_example <- plot_data_example[order(plot_data_example$Value),]
plot_scatter(plot_data_example,Filename = "")
```

plot_scatter_byFactors

Scatter plot with overlapping factors/groups.

Description

Represent different groups as defined by FACTOR(S) in a scatter plot.

Usage

```
plot_scatter_byFactors(MATRIX, FACTOR, factor_labs = NULL, xLab = NULL,
  yLab = NULL, Filename = "plot_scatter_byFactors.pdf", Width = 6,
  Height = 5)
```

Arguments

MATRIX	- two column matrix to plot. Only the first two columns will be used.
FACTOR	- character vector or list of character vectors used to split the data into groups.
factor_labs	- optional. Character vector to distinguish FACTOR groups. Default (NULL). See Details.
Filename	- optional. Character vector containing the file path to save the image. Must have an extension (see Details). Default ("plot_scatter_byFactors.pdf"). If set to "", no file will be written.
Width	- Width size for file. Default (12 in).
Height	- Height size for file. Default (9 in).

Details

This function is used by analysis_genomes_module_output() to plot the first two Principal Components (PCs) from the PCA analysis, overlapping different factors or groups (as determined by `FACTOR`). It is also used by analysis_pca_mean_distance_grouping() to highlight a single group on PC plot.

`factor_labs` is used as an extension for the filename for the plot files. The names of the `FACTOR` object OR "factor" followed by a number will be used if `factor_labs` is not specified (i.e. `factor_labs = NULL`).

A plot is only generated for FACTORS with less than 60 groups. All the data is plotted in grey in the background, with groups being overlayed for each Factor.

Only the following file types are allowed: pdf, png, svg and jpeg.

Value

A list with as many entires as factors is generated (one for each factor) using `factor_labs` for the names. For every `FACTOR`, a list will contain:

```
$nGroups - numeric. The number of groups found for that FACTOR
                    NOTE: That entries with "" will be excluded.
$table   - data frame with the number of entries for each group found.
$file    - character vector of file name(s).
```

See Also

analysis_genomes_module_output,analysis_pca_mean_distance_grouping

Examples

```
data(data_example_moduleIDs)
data(data_example_genomeIDs)
# length(data_example_genomeIDs) # [1] 25

# Calculate the module completion fraction (mcf) for the genomes and modules contained in the data objects above
OUT      <- query_genomes_to_modules(data_example_genomeIDs,MODULE_ID = data_example_moduleIDs)

pca <- prcomp(OUT$MATRIX)

# Make boxplots of the mcf output from query_genomes_to_modules
this_FACTOR <- rep(LETTERS[1:5],length(data_example_genomeIDs)/5)
plot_output <- plot_scatter_byFactors(pca$x[,1:2],FACTOR = this_FACTOR,factor_labs = "random",Filename = "plo

# NAs are ommitted, so a single group can be contrasted with overall data
this_FACTOR <- c(rep(NA,20),rep(LETTERS[1],5))
plot_output <- plot_scatter_byFactors(pca$x[,1:2],FACTOR = this_FACTOR,factor_labs = "group_A",Filename = "pl
```

`plot_sunburst` *Sunburst plot*

Description

This function arranges the hierarchical data and fills based on the counts for each category and level.

Usage

```
plot_sunburst(DATA, centerLabel = "", ANGLE = FALSE, fill_by = NULL,
  fill_by_mean = FALSE, outer_text = TRUE,
  outer_text_levelN_minus_1 = TRUE, legend_name = "Counts\n",
  sunburst = TRUE, setMax = NULL, setMin = NULL, Filename = "",
  WIDTH = 25, HEIGHT = 25)
```

Arguments

`DATA` - Data frame containing hierarchical data by columns, where the left column is the highest level and the right column the lowest one.

`centerLabel` - optional. String to be used for centre label. Default (`""`; i.e. no text).

`ANGLE` - optional. Should the angle of the text be adjusted to the position where it's at? Default (`FALSE`; i.e. horizontal text).

`fill_by` - optional. Numeric vector containing values to be used to fill outer most ring (see Details). Default (`NULL`).

`fill_by_mean` - optional. Should the categories in the inner rings be filled (colored) by the mean of the fill_by value provided. Default (`FALSE`).

`outer_text` - optional. Should the text from the lowest level (right column) be included? Default (`TRUE`).

`outer_text_levelN_minus_1`
- optional. Should the text from the second lowest level (second column from right) be included? Default (`TRUE`).

`legend_name` - optional. Default ("Counts\n"; i.e. have space between name and legend bar).

`sunburst` - optional. Should the sunburst be made (default) or should the output be bars? Default (`TRUE,`).

`setMax` - optional. Numerical value to set the maximum limit. Categories with higher values than those specified will be excluded, making them light blue. Default (`NULL`).

`setMin` - optional. Numerical value to set the minimun limit. Categories with lower values than those specified will be excluded, making them light blue. Default (`NULL`).

`Filename` - optional. A character vector containing the file path to save image(s). The device to save is determined by file extension. Default (`""`; i.e. printed to device, file not written).

`WIDTH` - optional. Width size for file. Note that text size in plot scales with `WIDTH`. Default (25 in).

`HEIGHT` - optional. Height size for file. Default (25 in).

Details

This function arranges a hierarchical data set into a dart-style chart, which is then partitioned and colored based on the data structure. The outer most section displays the number of elements (counts) in the lowest (most specific) hierarchical level. The user has the option to fill the outer level with user specified values given to `fill_by` or to fill with the mean value `fill_by_mean`. See `Details`.

The sunburst is generated by having a ring containing the hierarchical data from highest level (most generic) at the centre, followed by concentric rings that go out as the data becomes more specific. The outer-most ring is colored (filled) by the number of members (counts) in that category. The hierarchy is conserved throughout the data levels.

LEVEL_1	LEVEL_2	LEVEL_3
a	aa	aa1
a	aa	aa2
a	ab	ab1
a	ab	ab2
b	ba	ba1
b	ba	ba2
b	bb	bb1
b	bb	bb2
c	ca	ca1
c	ca	ca2
c	cb	cb1
c	cb	cb2

The above data would result in a sunburst plot that would have 3 categories in the inner-most ring, each with 2 categories in the second ring (6 total) and 12 categories in the outer ring, one for each of the second level categories. Therefore, the counts would all be 1 (see the first example for a case with different levels).

`fill_by` allows the user to display other values associated with the outer-most categories, such as the module fraction completeness (mfc), rather than the membership count. If `fill_by_mean` is set to `TRUE`, then the inner rings are also filled (colored) based on the mean of the `fill_by` value of the data making up the category.

`fill_by` should contain as many entries as there are categories at the last level. If more are provided, only the required entries will be used (i.e. if 5 values are provided for data with 3 categories, only the first 3 values will be used). Only as many entries as there are rows in `DATA` are allowed.

Function developed by expanding and modifying Yahia El Gamal's blog post "Create Basic Sunburst Graphs with ggplot2" ([https://medium.com/optima-blog/create-basic-sunburst-graphs-with-ggplot2-7d7484d92c61])

Value

List containing the `$DATA` used to generate the plot and the plot itself as a ggplot object `$p`.

Examples

```
data(data_example_sunburst)

#  CLASS_I        CLASS_II                                CLASS_III                NAME_SHORT
# Pathway module  Carbohydrate and lipid metabolism Fatty acid metabolism                           beta-Oxida
# Pathway module           Energy metabolism   Methane metabolism               Methanogenesis, from methano
# Pathway module           Energy metabolism   Methane metabolism                Methanogenesis, from acetate
# Pathway module           Energy metabolism     Carbon fixation Reductive acetyl-CoA pathway (Wood-Ljungdahl p
# Pathway module           Energy metabolism  Nitrogen metabolism              Dissimilatory nitrate reducti
# Pathway module           Energy metabolism   Methane metabolism             Methanogenesis, from methylami
# Pathway module           Energy metabolism   Methane metabolism                  Methanogenesis, from CO2
# Pathway module           Energy metabolism    Sulfur metabolism              Dissimilatory sulfate reductio

# Simplest plot using count data.
plot_data <- plot_sunburst(sunbust_data,WIDTH = 8) # WIDTH scales text size

# Specify values to be used for outer ring (lowest level) and change legend name accordingly
data(data_example_sunburst_fill_by) # Contains the mcf for the modules associated with the data shown above for
plot_data <- plot_sunburst(data_example_sunburst,centerLabel = "Org A",fill_by = data_example_sunburst_fill_b
# Also fill inner rings (levels) according to the mean values determined by fill_by.
plot_data <- plot_sunburst(data_example_sunburst,centerLabel = "Org A",fill_by = data_example_sunburst_fill_b
```

plot_variance_boxplot *Make a boxplot*

Description

Make a boxplot

Usage

```
plot_variance_boxplot(MATRIX_IN, x_lab = "Modules",
  y_lab = "Fraction matched", xLabs_angle = T, Filename = "",
  Width = 24, Height = 18)
```

Arguments

MATRIX_IN - numeric matrix containing the module fraction match of the genomes (rows) and modules (columns). The columns are used as factor for the box plot grouping.

x_lab - optional. String to specify the x-axis label corresponding to the row values. Default ("Modules").

y_lab - optional. String to specify the legend title. Default ("Fraction\n matched").

xLabs_angle - optional. Should x-axis labels be rotated 45 degrees? Default (TRUE).

Filename - optional. A character vector containing the file path to save image(s). The device to save is determined by file extension. Default ("", i.e. file not written).

Width - Width size for file. Default (24in).

Height - Height size for file. Default (18in).

Value

ggplot object of plot

query_genes_to_genomes *Find the genome(s) that contain a set of genes.*

Description

The genes can be either an enzyme, given by its Enzyme Classification (EC) number (e.g. "1.1.1.1"), or a KEGG ortholog identifier (K number, e.g. "K00001").

Usage

```
query_genes_to_genomes(genes, use_genome_reference_table = NULL,
  use_ko_reference_table = NULL, use_enzyme_reference_table = NULL)
```

Arguments

`genes` - character vector of KO or EC numbers. See Details for more info on the use of ECs.

`use_genome_reference_table`
- optional. Provide a data frame with updated KEGG genome database. Default (`NULL`; inbuilt data used). See Details.

`use_ko_reference_table`
- optional. Provide a data frame with updated KEGG ortholog database. Default (`NULL`; inbuilt data used). See Details.

`use_enzyme_reference_table`
- optional. Provide a data frame with updated KEGG enzyme database. Default (`NULL`; inbuilt data used). See Details.

Details

When EC numbers are used, at least the first three positions must be defined (e.g. "1.1.1" and "1.1.1.-" are both allowed, but "1.1.-" or "1.1.-.-" would not be allowed and an warning message will be displayed). If incomplete or undefined EC numbers are give, the function will automatically add the enzymes for that class (e.g. "1.1.1.1", "1.1.1.10", etc).

The `use_` set of arguments allow users with KEGG FTP access to provide the updated data from the KEGG databases in the form of reference tables AND/OR for advanced users to provide custom-made modules (see below). These reference tables can be generated with the `parseKEGG` family of functions and need to have a specific format (see function descriptions for details on format).

If providing `use_genome_reference_table`, make sure that the `parseKEGG_genome` function is run with arguments `addECs = T` and/or `addKOs = T` to include genes that make up the genomes.

`ko_reference_table` and `enzyme_reference_table` are used to verify that the `genes` provided exist within the KEGG database.

Value

Data frame containing the `genes` (rows, specified in the rownames) and the T number of the KEGG genomes (columns) with a binary indicator for presence of gene in given genome.

See Also

parseKEGG_genome, ko_reference_table, enzyme_reference_table

Examples

```
genomes <- query_genes_to_genomes("K00844")

genomes <- query_genes_to_genomes("1.1.1.1")

genomes <- query_genes_to_genomes(genes = paste("K0000",1:3,sep=""))
```

query_genes_to_modules

Given a set of genes, find the modules they are involved in.

Description

The genes can be either enzymes, given by its Enzyme Classification (EC) number (e.g. "1.1.1.1"), or KEGG ortholog identifiers (K number, e.g. "K00001"). Using EC or K numbers might give different results, as an EC number might map to multiple K numbers or none.

Usage

```
query_genes_to_modules(genes, use_module_reference_table = NULL,
  use_ko_reference_table = NULL, use_enzyme_reference_table = NULL)
```

Arguments

`genes`	- character vector (length 1). K or EC number.
`use_module_reference_table`	- optional. Provide a data frame with updated KEGG module database OR with custom-made modules. Default (`NULL`; inbuilt data used). See Details.
`use_ko_reference_table`	- optional. Provide a data frame with updated KEGG ortholog database. Default (`NULL`; inbuilt data used). See Details.
`use_enzyme_reference_table`	- optional. Provide a data frame with updated KEGG enzyme database. Default (`NULL`; inbuilt data used). See Details.

Details

The `use_` set of arguments allow users with KEGG FTP access to provide the updated data from the KEGG databases in the form of reference tables AND/OR for advanced users to provide custom-made modules (see below). These reference tables can be generated with the `parseKEGG` family of functions and need to have a specific format (see function descriptions for details on format).

The module definition (contained in `module_reference_table`) describes the relationship between genes and modules and is used to identify the modules in which `genes` is involved. The user can provide custom-made module definitions that use the logical expression format (however, the table format must be conserved!).

If enzyme identifiers are provided, note that as there are lingering unspecified enzymes (e.g. '1.1.1.-') in the KEGG module EC-based definition, this function also returns a row entry for the unspecified-versions of enzymes provided involved in one or more modules.

Value

Data frame containing a binary indicator for `genes` (rows, specified by rownames) involved in modules (columns, Module IDs specified in names). A column called '`no_match`' is returned if one or more of the genes is not involved in any modules.

`NULL` is returned when there are no valid entries to evaluate. Note that enzyme entries are checked for 4 position completeness.

See Also

parseKEGG_module

Examples

```
modules <- query_genes_to_modules("K00844")

modules <- query_genes_to_modules("1.1.1.1")
```

---

query_genomes_to_modules

Evaluates the KEGG modules presence given genome information.

Description

This function returns the 'completeness' of KEGG modules in the provided genome(s) (either as a genome identifier or as a lists of Enzyme Comission (EC) number or KEGG ortholog identifier, K number). The user can define which modules the function should check by providing a single or set of modules under 'MODULE_ID'. If this is left blank, the function returns completeness of all KEGG modules (excluding modules defined in terms of other modules). The function can also be restricted to a subset of all KEGG modules by using the SEARCH_NAME and SEARCH_CLASS arguments. See Details.

Usage

```
query_genomes_to_modules(GENOME_INFO, splitBy = "[;]", GENOME_ID_COL = 1,
  GENES_COL = 2, MODULE_ID = "", SEARCH_NAME = "", SEARCH_CLASS_I = "",
  SEARCH_CLASS_II = "", SEARCH_CLASS_III = "", EXCLUDE_NAME = "",
  OUT_MODULE_NAME = FALSE, META_OUT = FALSE, ADD_OUT = FALSE,
  use_genome_reference_table = NULL, ...)
```

Arguments

GENOME_INFO - character vector containing genome identifier(s) or organism name(s) OR data frame containing genome identifier/names(s) and gene list. See Details.

splitBy - string indicating the split pattern for the data contained in the column indicated by GENES_COL. Default (" [;] ": uses ' ; ' , the ' [ ] ' indicates that it is NOT a regular expression).

GENOME_ID_COL - optional. Column NAME or NUMBER containing genome NAME or IDENTIFIER. Default (1; <first column>). See Details.

GENES_COL - optional. Column NAME or NUMBER pointing to the GENES. Default (2; <second column>). See Details.

MODULE_ID - optional. Character vector listing specific KEGG module IDs (e.g. "M00001"). Default ("").

SEARCH_NAME - optional. Character vector listing terms to search in KEGG module NAME field (case-insensitive). Default ("").

SEARCH_CLASS_I - optional. Character vector listing terms to search in KEGG module CLASS_I field (case-insensitive). Default ("").

SEARCH_CLASS_II
: - optional. Character vector listing terms to search in KEGG module CLASS_II field (case-insensitive). Default ("").

SEARCH_CLASS_III
: - optional. Character vector listing terms to search in KEGG module CLASS_III field (case-insensitive). Default ("").

EXCLUDE_NAME - optional. Character vector listing terms that if matched in KEGG module NAME field will be excluded (case-insensitive). Default ("").

OUT_MODULE_NAME
: - optional (logical). Should the column names of MATRIX be the module IDs (M numbers) or module names? Default (FALSE; return matrix with M numbers)

META_OUT - optional (logical). Should the KEGG module metadata be outputed? Default (FALSE).

ADD_OUT - optional (logical). Should additional information be outputed? Default (FALSE).

use_genome_reference_table
: - optional. Provide a data frame with updated KEGG module database. Default (NULL; inbuilt data used). See Details.

... - further arguments passed to misc_geneVector_module, such as KO_in_DEF_EC or use_module_reference_table. See Details.

Details

This function processes the GENOME_INFO by pasing each in turn to misc_geneVector_module() and collating all the output. GENOME_ID_COL and GENES_COL are only used if GENOME_INFO is a data frame. Post-analysis of this output can be carried out with analysis_genomes_module_output().

Appropriate GENOME_INFO input can be either a character vector or a data frame:

```
*character vector*
   options:
     (1) KEGG taxonomy identifier (T0 number; e.g. "T00001")
         AND/OR KEGG organism identifier (3 or 4 letter code; e.g. "eco").
     (2) Organism's scientific name (e.g. "Escherichia coli"; case-insensitive).
           Multiple strains might be matched in the KEGG organisms database and
          all will be processed. Strain information or full organism name can be
          added to reduce the search results; use the 'Definition' entry in KEGG
           \url{http://www.genome.jp/kegg/catalog/org_list.html}.

         NOTE: do NOT combine NAMES with IDENTIFIERS
         WARNING issued when there is no matching identifier in the
                  KEGG genome/organism databases.

*data frame*
   > column I  - genome name/identifier (pointed to by \code{GENOME_ID_COL}).
  > column II - concatenated string of either EC or K numbers using \code{splitBy}
                  as delimiter (pointed to by \code{GENES_COL}).
```

KO_in_DEF_EC - If the genes given are EC numbers, should K numbers present in the KEGG module definition be assumed to be present or not? Default (FALSE).

The use_ set of argument allows users with KEGG FTP access to provide the updated data from the KEGG databases in the form of reference tables. These reference tables can be generated with the parseKEGG family of functions and need to have a specific format (see function descriptions for details on format).

Value

Returns a list containing the following objects:

`$MATRIX` - matrix of the datasets (rows) and the modules searched (columns) containing the fraction completeness.

`$QUERIES` - data frame listing the SEARCH_TERMS and ARGUMENTS used.

`$METADATA` - data frame containing the metadata from the modules analysed (if `META_OUT == TRUE`).

```
Columns:
 (1) MODULE_ID     (4) CLASS_I     (7) DEFINITION
 (2) MODULE_NAME   (5) CLASS_II    (8) OPTIONAL
 (3) NAME_SHORT    (6) CLASS_III
```

An OPTIONAL entry of 'NA' indicates that there are no optional K numbers in that module.

`$ADD_INFO` - data frame containing additional information of the analysis (if `ADD_OUT == TRUE`).

```
Columns:
 (1) GENOME_ID     (3) NAME_SHORT   (5) nBLOCKS    (7) OPTIONAL_PRESENT
 (2) MODULE_ID     (4) FRACTION     (6) COVERAGE
```

where 'COVERAGE' refers to the genes provided that are involved in the given module and genome.

See misc_geneVector_module for additional details on the output.

See Also

misc_geneVector_module, analysis_genomes_module_output, plot_heatmap, data_example_multi_ECs_KOs

Examples

```
## USE T numbers
T_NUMEBERS <- paste("T0000",1:5,sep="")
OUT <- query_genomes_to_modules(T_NUMEBERS,MODULE_ID = paste("M0000",1:5,sep=""),
                                 META_OUT = T, ADD_OUT = T)

## USE SPECIES NAMES
names <- c("escherichia coli","heliobacter")
OUT <- query_genomes_to_modules(names,MODULE_ID = paste("M0000",1:5,sep=""),
                                 META_OUT = T, ADD_OUT = T)

## USE USER-SPECIFIED GENE SETS
data(data_example_multi_ECs_KOs) # load example data set
names(data_example_multi_ECs_KOs)
   # "ID"       "ORG_ID"   "ORGANISM" "KOs"      "ECs"
OUT <- query_genomes_to_modules(data_example_multi_ECs_KOs,GENOME_ID_COL = "ID",
                                GENES_COL = "KOs",MODULE_ID = paste("M0000",1:5,sep=""),
                                 META_OUT = T,ADD_OUT = T)

# Using EC numbers (less accurate)
OUT <- query_genomes_to_modules(data_example_multi_ECs_KOs,GENOME_ID_COL = "ID",
                                GENES_COL = "ECs",MODULE_ID = paste("M0000",1:5,sep=""),
                                 META_OUT = T,ADD_OUT = T)
```

query_missingGenes_from_module

Identify missing genes from a KEGG module given a set of genes.

Description

Identify missing genes from a KEGG module given a set of genes or a genome ID to obtain a complete module.

Usage

```
query_missingGenes_from_module(GENOME, MODULE_ID, PRINT_TO_SCREEN = TRUE,
  use_genome_reference_table = NULL, use_module_reference_table = NULL)
```

Arguments

`GENOME` - character vector containing a single genome identifier or set of genes or enzymes that define a [meta]genome. See Details.

`MODULE_ID` - KEGG module ID to be analysed (e.g. `"M00001"`).

`PRINT_TO_SCREEN` - logical. Should a print-friendly result be displayed? Default(`TRUE`).

`use_genome_reference_table` - optional. Provide a data frame with updated KEGG genome database. Default (`NULL`; inbuilt data used). See Details.

`use_module_reference_table` - optional. Provide a data frame with updated KEGG module database OR with custom-made modules. Default (`NULL`; inbuilt data used). See Details.

Details

`GENOME` can be a genome identifier (T0 number or a 3 or 4 letter code; e.g. "T00001" or "eco", respectively) OR a character vector contianing a set of genes (i.e. either EC or K numbers; e.g. "1.1.1.1" or "K00001", respectively). Examples of the latter can be found in the following objects: data_example_KOnumbers_vector or data_example_ECnumbers_vector.

Organism name NOT supported. Use the KEGG database website (`http://www.genome.jp/kegg/catalog/org_list.html`) to determine the genome identifier.

Note that the pipe ('|') in the DEFINITION indicates an OR operation This means that there are several possibile genes that carry out the same reaction or function and only one is required.

The `use_` set of arguments allow users with KEGG FTP access to provide the updated data from the KEGG databases in the form of reference tables AND/OR for advanced users to provide custom-made modules (see below). These reference tables can be generated with the `parseKEGG` family of functions and need to have a specific format (see function descriptions for details on format).

The module definition (contained in `module_reference_table`) describes the relationship between genes and modules and is used to identify the modules in which gene is involved. The user can provide custom-made module definitions that use the logical expression format (however, the table format must be conserved!).

Value

Data frame containing the following columns:

```
BLOCK_DEF     - the KEGG module DEFINITION of each block,
                  with missing genes flagged by '*';
PRESENT       - binary indicator of the automatic evaluation;
MISSING_GENES - list of missing genes.
```

Examples

```
# Load data
data("data_example_KOnumbers_vector")
OUT <- query_missingGenes_from_module(data_example_KOnumbers_vector,"M00001")
```

---

query_modules_to_genomes

Find a KEGG genome that has a given KEGG module complete

Description

Find a KEGG genome that has a given KEGG module complete (to a `threshold`). The module 'completeness' is based on the fraction of complete module blocks (given the KEGG module logical definition). Thus, a threshold of 0.5 would mean that the function would return all genomes that contain at least half of the blocks of the given module. See parseKEGG_module for further details on the KEGG module database.

Usage

```
query_modules_to_genomes(MODULE_ID, threshold = 1,
  use_matrix_dataframe = NULL, use_module_reference_table = NULL)
```

Arguments

`MODULE_ID` - KEGG module identifier (M number, e.g. "M00001").

`threshold` - optional. Completness fraction desired (greater or equal). Default (1). Used as `fraction>=threshold`.

`use_matrix_dataframe` - optional. Provide module completeness fraction matrix or data frame of all KEGG genome entries with updated data. Default (`NULL`; inbuilt data used). See Details.

`use_module_reference_table` - optional. Provide a data frame with updated KEGG module database OR with custom-made modules. Default (`NULL`; inbuilt data used). See Details.

Format

`use_matrix_dataframe` rows - genome identifiers, columns - module IDs.

Details

The use_ set of argument allows users with KEGG FTP access to provide the updated data from the KEGG databases in the form of reference tables AND/OR for advanced users to provide custom-made modules (see below). These reference tables can be generated with the parseKEGG family of functions and need to have a specific format (see function descriptions for details on format).

To generate use_matrix_dataframe, store the $MATRIX output of query_genomes_to_modules when providing all genomes using either the module default (i.e. all modules) or specifying a subset of modules AND/OR with custom-made module definitions.

Value

If a single MODULE_ID is provided, a vector which contains the mfc with the KEGG genome identifiers (T number) as names is returned.

If multiple MODULE_IDs are provided, a matrix containing the mfc with the KEGG genome identifiers (T number) as row names and the module IDs as column names is returned.

See Also

parseKEGG_module, query_genomes_to_modules

Examples

```
genomes <- query_modules_to_genomes("M00001")

genomes <- query_modules_to_genomes(MODULE_ID = c("M00001","M00002"),threshold = 0.9)
```

Index