Package ‘ggkegg’

May 29, 2024

Type Package
Title KEGG pathway visualization by ggplot2
Version 1.2.0
Description This package aims to import, parse, and analyze KEGG data such as KEGG PATHWAY and KEGG MODULE. The package supports visualizing KEGG information using ggplot2 and ggraph through using the grammar of graphics. The package enables the direct visualization of the results from various omics analysis packages.
License MIT + file LICENSE
Encoding UTF-8
Depends R (>= 4.3.0), ggplot2, ggraph, XML, igraph, tidygraph
Imports BiocFileCache, GetoptLong, data.table, dplyr, magick, patchwork, shadowtext, stringr, tibble, org.Hs.eg.db, methods, utils, stats, AnnotationDbi, grDevices, gtable
Suggests knitr, clusterProfiler, bnlearn, rmarkdown, BiocStyle, testthat (>= 3.0.0)
RoxygenNote 7.3.0
biocViews Pathways, DataImport, KEGG
VignetteBuilder knitr
URL https://github.com/noriakis/ggkegg
BugReports https://github.com/noriakis/ggkegg/issues
Config/testthat/edition 3
git_url https://git.bioconductor.org/packages/ggkegg
git_branch RELEASE_3_19
git_last_commit 42a88d0
git_last_commit_date 2024-04-30
Repository Bioconductor 3.19
Date/Publication 2024-05-29
Author Noriaki Sato [cre, aut]
Maintainer Noriaki Sato <nori@hgc.jp>
## Contents

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>add_title</td>
<td>3</td>
</tr>
<tr>
<td>append_cp</td>
<td>4</td>
</tr>
<tr>
<td>append_label_position</td>
<td>4</td>
</tr>
<tr>
<td>assign_deseq2</td>
<td>5</td>
</tr>
<tr>
<td>carrow</td>
<td>6</td>
</tr>
<tr>
<td>combine_with_bnlearn</td>
<td>7</td>
</tr>
<tr>
<td>convert_id</td>
<td>7</td>
</tr>
<tr>
<td>create_test_module</td>
<td>9</td>
</tr>
<tr>
<td>create_test_network</td>
<td>9</td>
</tr>
<tr>
<td>create_test_pathway</td>
<td>10</td>
</tr>
<tr>
<td>edge_matrix</td>
<td>10</td>
</tr>
<tr>
<td>edge_numeric</td>
<td>11</td>
</tr>
<tr>
<td>edge_numeric_sum</td>
<td>12</td>
</tr>
<tr>
<td>geom_kegg</td>
<td>13</td>
</tr>
<tr>
<td>geom_node_rect</td>
<td>13</td>
</tr>
<tr>
<td>geom_node_rect_kegg</td>
<td>14</td>
</tr>
<tr>
<td>geom_node_shadowtext</td>
<td>15</td>
</tr>
<tr>
<td>get_module_attribute</td>
<td>16</td>
</tr>
<tr>
<td>get_module_attribute,kegg_module-method</td>
<td>16</td>
</tr>
<tr>
<td>get_network_attribute</td>
<td>17</td>
</tr>
<tr>
<td>get_network_attribute,kegg_network-method</td>
<td>17</td>
</tr>
<tr>
<td>ggkegg</td>
<td>18</td>
</tr>
<tr>
<td>ggkeggsave</td>
<td>19</td>
</tr>
<tr>
<td>ggplot_add.geom_kegg</td>
<td>20</td>
</tr>
<tr>
<td>ggplot_add.geom_node_rect_kegg</td>
<td>20</td>
</tr>
<tr>
<td>ggplot_add.overlay_raw_map</td>
<td>21</td>
</tr>
<tr>
<td>ggplot_add.stamp</td>
<td>22</td>
</tr>
<tr>
<td>highlight_entities</td>
<td>22</td>
</tr>
<tr>
<td>highlight_module</td>
<td>23</td>
</tr>
<tr>
<td>highlight_set_edges</td>
<td>24</td>
</tr>
<tr>
<td>highlight_set_nodes</td>
<td>25</td>
</tr>
<tr>
<td>module</td>
<td>26</td>
</tr>
<tr>
<td>module_abundance</td>
<td>26</td>
</tr>
<tr>
<td>module_completeness</td>
<td>27</td>
</tr>
<tr>
<td>module_text</td>
<td>28</td>
</tr>
<tr>
<td>multi_pathway_native</td>
<td>28</td>
</tr>
<tr>
<td>network</td>
<td>29</td>
</tr>
<tr>
<td>network_graph</td>
<td>30</td>
</tr>
<tr>
<td>node_matrix</td>
<td>30</td>
</tr>
<tr>
<td>node_numeric</td>
<td>31</td>
</tr>
<tr>
<td>obtain_sequential_module_definition</td>
<td>32</td>
</tr>
<tr>
<td>output_overlay_image</td>
<td>32</td>
</tr>
<tr>
<td>overlay_raw_map</td>
<td>34</td>
</tr>
<tr>
<td>pathway</td>
<td>35</td>
</tr>
<tr>
<td>pathway_abundance</td>
<td>36</td>
</tr>
<tr>
<td>pathway_info</td>
<td>37</td>
</tr>
</tbody>
</table>
add_title

Description
Add the title to the image produced by output_overlay_image using magick.

Usage
add_title(out, title = NULL, size = 20, height = 30, color = "white", titleColor = "black", gravity = "west")

Arguments
out the image
title the title
size the size
height title height
color bg color
titleColor title color
gravity positioning of the title in the blank image

Value
output the image
Description

append clusterProfiler results to graph

Usage

append_cp(res, how = "any", name = "name", pid = NULL, infer = FALSE)

Arguments

- **res**: enrichResult class
- **how**: how to determine whether the nodes is in enrichment results
- **name**: name column to search for query
- **pid**: pathway ID, if NULL, try to infer from graph attribute
- **infer**: if TRUE, append the prefix to queried IDs based on pathway ID

Value

enrich_attribute column in node

Examples

```r
graph <- create_test_pathway()
nodes <- graph |> data.frame()
if (require("clusterProfiler")) {
  cp <- enrichKEGG(nodes$name |>
    strsplit(quote(\:)) |>
    vapply("\[", 2, FUN.VALUE="character"))
  ## This append graph node logical value whether the
  ## enriched genes are in pathway
  graph <- graph |> mutate(cp=append_cp(cp, pid="hsa05322"))
}
```

Description

Append the label position at center of edges in global map like ko01100 where line type nodes are present in KGML. Add 'center' column to graph edge.
assign_deseq2

Usage

append_label_position(g)

Arguments

g graph

Value
tbl_graph

Examples

## Simulate nodes containing `graphics_type` of line and `coords`

```r
gm_test <- data.frame(name="ko:K00112",type="ortholog",reaction="rn:R00112",
graphics_name="K00112",fgcolor="#ff0000",bgcolor="#ffffff",
graphics_type="line",coords="1,2,3,4",orig.id=1,pathway_id="test")
gm_test <- tbl_graph(gm_test)
test <- process_line(gm_test) |> append_label_position()
```

---

assign_deseq2 assign_deseq2

Description

assign DESeq2 numerical values to nodes

Usage

```r
assign_deseq2(
  res,
  column = "log2FoldChange",
  gene_type = "SYMBOL",
  org_db = org.Hs.eg.db,
  org = "hsa",
  numeric_combine = mean,
  name = "name"
)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>res</td>
<td>The result() of DESeq()</td>
</tr>
<tr>
<td>column</td>
<td>column of the numeric attribute, default to log2FoldChange</td>
</tr>
<tr>
<td>gene_type</td>
<td>default to SYMBOL</td>
</tr>
<tr>
<td>org_db</td>
<td>organism database to convert ID to ENTREZID</td>
</tr>
<tr>
<td>org</td>
<td>organism ID in KEGG</td>
</tr>
</tbody>
</table>
numeric_combine

how to combine multiple numeric values

name

column name for ID in tbl_graph nodes

Value

numeric vector

Examples

```r
graph <- create_test_pathway()
res <- data.frame(row.names="6737", log2FoldChange=1.2)
graph <- graph |> mutate(num=assign_deseq2(res, gene_type="ENTREZID"))
```
combine_with_bnlearn

Description

combine the reference KEGG pathway graph with bnlearn boot.strength output

Usage

combine_with_bnlearn(pg, str, av, prefix = "ko:", how = "any")

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pg</td>
<td>reference graph (output of 'pathway')</td>
</tr>
<tr>
<td>str</td>
<td>strength data.frame</td>
</tr>
<tr>
<td>av</td>
<td>averaged network to plot</td>
</tr>
<tr>
<td>prefix</td>
<td>add prefix to node name of original averaged network like, ‘hsa:’ or ‘ko:’.</td>
</tr>
<tr>
<td>how</td>
<td>‘any’ or ‘all’</td>
</tr>
</tbody>
</table>

Value

tbl_graph

Examples

if (requireNamespace("bnlearn", quietly=TRUE)) {
  ## Simulating boot.strength() results
  av <- bnlearn::model2network("[6737|51428][51428]")
  str <- data.frame(from="51428",to="6737",strength=0.8,direction=0.7)
  graph <- create_test_pathway()
  combined <- combine_with_bnlearn(graph, str, av, prefix="hsa:")
}

convert_id

Description

convert the identifier using retrieved information
Usage

convert_id(
  org,
  name = "name",
  convert_column = NULL,
  colon = TRUE,
  first_arg_comma = TRUE,
  sep = " ",
  first_arg_sep = TRUE,
  divide_semicolon = TRUE,
  edge = FALSE
)

Arguments

org which identifier to convert
name which column to convert in edge or node table
convert_column which column is parsed in obtained data frame from KEGG REST API
colon whether the original ids include colon (e.g. ‘ko:’) If ‘NULL’, automatically set according to ‘org’
first_arg_comma take first argument of comma-separated string, otherwise fetch all strings
sep separator to separate node names, default to space
first_arg_sep take first argument if multiple identifiers are in the node name, otherwise parse all identifiers
divide_semicolon whether to divide string by semicolon, and take the first value
divide_semicolon if converting edges

Value

vector containing converted IDs

Examples

graph <- create_test_pathway()
graph <- graph |> mutate(conv=convert_id("hsa"))
create_test_module

Description
Test kegg_module for examples and vignettes. The module has no biological meanings.

Usage
create_test_module()

Value
return a test module to use in examples

Examples
create_test_module()

create_test_network

Description
create_test_network

Usage
create_test_network()

Value
test network

Examples
create_test_network()
create_test_pathway

Description

As downloading from KEGG API is not desirable in vignettes or examples, return the ‘tbl_graph’ with two nodes and two edges.

Usage

create_test_pathway(line = FALSE)

Arguments

line return example containing graphics type line

Value
tbl_graph

Examples

create_test_pathway()

edge_matrix

description

given the matrix representing gene as row and sample as column, append the edge value (sum of values of connecting nodes) to edge matrix and return tbl_graph object. The implementation is based on the paper by Adnan et al. 2020 (https://doi.org/10.1186/s12859-020-03692-2).

Usage

describe(matrix,
    graph,
    mat,
    gene_type = "SYMBOL",
    org = "hsa",
    org_db = org.Hs.eg.db,
    num_combine = mean
)
edge_numeric

Arguments
- **graph**: tbl_graph to append values to
- **mat**: matrix representing gene as row and sample as column
- **gene_type**: gene ID of matrix row
- **org**: organism ID to convert ID
- **org_db**: organism database to convert ID
- **num_combine**: function to combine multiple numeric values

Value
tbl_graph

Examples
```r
graph <- create_test_pathway()
num_df <- data.frame(row.names=c("6737","51428"),
                     "sample1"=c(1.1,1.2),
                     "sample2"=c(1.1,1.2),
                     check.names=FALSE)
graph <- graph |> edge_matrix(num_df, gene_type="ENTREZID")
```

Description
add numeric attribute to edge of tbl_graph

Usage
```r
dge_numeric(num, num_combine = mean, how = "any", name = "name")
```

Arguments
- **num**: named vector or tibble with id and value column
- **num_combine**: how to combine number when multiple hit in the same node
- **how**: 'any' or 'all'
- **name**: name of column to match for

Value
numeric vector
**Examples**

```r
graph <- create_test_pathway()
graph <- graph |> activate("edges") |> mutate(num=edge_numeric(c(1.1) |> setNames("degradation"), name="subtype_name"))
```

---

**Description**

add numeric attribute to edge of tbl_graph based on node values. The implementation is based on the paper by Adnan et al. 2020 (https://doi.org/10.1186/s12859-020-03692-2).

**Usage**

```r
edge_numeric_sum(num, num_combine = mean, how = "any", name = "name")
```

**Arguments**

- **num**: named vector or tibble with id and value column
- **num_combine**: how to combine number when multiple hit in the same node
- **how**: ‘any’ or ‘all’
- **name**: name of column to match for

**Value**

numeric vector

**Examples**

```r
graph <- create_test_pathway()
graph <- graph |> activate("edges") |> mutate(num=edge_numeric_sum(c(1.2,-1.2) |> setNames(c("TRIM21","DDX41")), name="graphics_name"))
```
**geom_kegg**

Description

Wrapper function for plotting KEGG pathway graph add geom_node_rect, geom_node_text and geom_edge_link simultaneously

Usage

```r
geom_kegg(
  edge_color = NULL,
  node_label = .data$name,
  group_color = "red",
  parallel = FALSE
)
```

Arguments

- `edge_color` color attribute to edge
- `node_label` column name for node label
- `group_color` border color for group node rectangles
- `parallel` use geom_edge_parallel() instead of geom_edge_link()

Value

ggplot2 object

Examples

```r
test_pathway <- create_test_pathway()
p <- ggraph(test_pathway, layout="manual", x=x, y=y)+
  geom_kegg()
```

**geom_node_rect**

Description

Plot rectangular shapes to ggplot2 using GeomRect, using StatFilter in ggraph
Usage

```r
geom_node_rect(
  mapping = NULL,
  data = NULL,
  position = "identity",
  show.legend = NA,
  ...
)
```

Arguments

- `mapping`: aes mapping
- `data`: data to plot
- `position`: positional argument
- `show.legend`: whether to show legend
- `...`: passed to ‘params‘ in ‘layer()‘ function

Value

- `geom`

Examples

```r
test_pathway <- create_test_pathway()
plt <- ggraph(test_pathway, layout="manual", x=x, y=y) +
  geom_node_rect()
```

Description

Wrapper function for plotting a certain type of nodes with background color with `geom_node_rect()`

Usage

```r
geom_node_rect_kegg(type = NULL, rect_fill = "grey")
```

Arguments

- `type`: type to be plotted (gene, map, compound ...)
- `rect_fill`: rectangular fill

Value

- `ggplot2 object`
**geom_node_shadowtext**

**Examples**

```r
test_pathway <- create_test_pathway()
plt <- ggraph(test_pathway, layout="manual", x=x, y=y) +
    geom_node_rect_kegg(type="gene")
```

**Description**

Plot shadowtext at node position, use StatFilter in ggraph

**Usage**

```r
geom_node_shadowtext(
    mapping = NULL,
    data = NULL,
    position = "identity",
    show.legend = NA,
    ...
)
```

**Arguments**

- `mapping`: aes mapping
- `data`: data to plot
- `position`: positional argument
- `show.legend`: whether to show legend
- `...`: passed to `params` in `layer()` function

**Value**

geom

**Examples**

```r
test_pathway <- create_test_pathway()
plt <- ggraph(test_pathway, layout="manual", x=x, y=y) +
    geom_node_shadowtext(aes(label=name))
```
**Description**
get slot from 'kegg_module' class

**Usage**
```r
get_module_attribute(x, attribute)
```

**Arguments**
- `x`: kegg_module class object
- `attribute`: pass to get_module_attribute

**Value**
attribute of kegg_module

**Description**
get the kegg_module class attribute

**Usage**
```r
## S4 method for signature 'kegg_module'
get_module_attribute(x, attribute)
```

**Arguments**
- `x`: kegg_module class object
- `attribute`: slot name

**Value**
attribute of kegg_module
get_network_attribute

Description
get slot from 'kegg_network' class

Usage
get_network_attribute(x, attribute)

Arguments

x kegg_network class object
attribute pass to get_network_attribute

Value
attribute of kegg_network

Description
get the kegg_network class attribute

Usage
## S4 method for signature 'kegg_network'
get_network_attribute(x, attribute)

Arguments

x kegg_network class object
attribute slot name

Value
attribute of kegg_module
Description

main function parsing KEGG pathway data, making igraph object and passing it to ggraph.

Usage

```r
ggkegg(
  pid,
  layout = "native",
  return_igraph = FALSE,
  return_tbl_graph = FALSE,
  pathway_number = 1,
  convert_org = NULL,
  convert_first = TRUE,
  convertCollapse = NULL,
  convert_reaction = FALSE,
  delete_undefined = FALSE,
  delete_zero_degree = FALSE,
  numeric_attribute = NULL,
  node_rect_nudge = 0,
  group_rect_nudge = 2,
  module_type = "definition",
  module_definition_type = "text"
)
```

Arguments

- **pid**: KEGG Pathway id e.g. hsa04110
- **layout**: default to "native", using KGML positions
- **return_igraph**: return the resulting igraph object
- **return_tbl_graph**: return the resulting tbl_graph object (override 'return_igraph' argument)
- **pathway_number**: pathway number if passing enrichResult
- **convert_org**: these organism names are fetched from REST API and cached, and used to convert the KEGG identifiers. e.g. c("hsa", "compound")
- **convert_first**: after converting, take the first element as node name when multiple genes are listed in the node
- **convertCollapse**: if not NULL, collapse the gene names by this character when multiple genes are listed in the node.
- **convert_reaction**: reaction name (graph attribute ‘reaction’) will be converted to reaction formula
delete_undefined
delete ‘undefined’ node specifying group, should be set to ‘TRUE’ when the
layout is not from native KGML.
delete_zero_degree
delete nodes with zero degree, default to FALSE
numeric_attribute
named vector for appending numeric attribute
node_rect_nudge
parameter for nudging the node rect
group_rect_nudge
parameter for nudging the group node rect
module_type
specify which module attributes to obtain (definition or reaction)
module_definition_type
‘text’ or ‘network’ when parsing module definition. If ‘text’, return ggplot ob-
ject. If ‘network’, return ‘tbl_graph’.

Value
ggplot2 object

Examples

## Use pathway ID to obtain `ggraph` object directly.
g <- ggkegg("hsa04110")
g + geom_node_rect()

Usage
ggkeggsave(filename, plot, dpi = 300, wscale = 90, hscale = 90)

Arguments

filename file name of the image
plot plot to be saved
dpi dpi, passed to ggsave
wscale width scaling factor for pixel to inches
hscale height scaling factor for pixel to inches
Value

save the image

Description

ggplot_add.geom_kegg

Usage

## S3 method for class 'geom_kegg'
ggplot_add(object, plot, object_name)

Arguments

object An object to add to the plot
plot The ggplot object to add object to
object_name The name of the object to add

Value

ggplot2 object

Examples

test_pathway <- create_test_pathway()
p <- ggraph(test_pathway, layout="manual", x=x, y=y)+
  geom_kegg()
Arguments

- **object**: An object to add to the plot
- **plot**: The ggplot object to add object to
- **object_name**: The name of the object to add

Value

ggplot2 object

Examples

test_pathway <- create_test_pathway()
plt <- ggraph(test_pathway, layout="manual", x=x, y=y) +
geom_node_rect_kegg(type="gene")

## Need `pathway_id` column in graph
## if the function is to automatically infer
graph <- create_test_pathway() |> mutate(pathway_id="hsa04110")
ggraph(graph) + overlay_raw_map()
Description

ggplot_add.stamp

Usage

## S3 method for class 'stamp'
ggplot_add(object, plot, object_name)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>An object to add to the plot</td>
</tr>
<tr>
<td>plot</td>
<td>The ggplot object to add object to</td>
</tr>
<tr>
<td>object_name</td>
<td>The name of the object to add</td>
</tr>
</tbody>
</table>

Value

ggplot2 object

Examples

test_pathway <- create_test_pathway()
plt <- ggraph(test_pathway, layout="manual", x=x, y=y) +
      stamp("hsa:6737")

Description

highlight the entities in the pathway, overlay raw map and return the results. Note that highlighted
nodes are considered to be rectangular, so it is not compatible with the type like 'compound'.

Usage

highlight_entities(
  pathway,
  set,
  how = "any",
  num_combine = mean,
  name = "graphics_name",
  sep = ","
highlight_module

```r
no_sep = FALSE,
show_type = "gene",
fill_color = "tomato",
legend_name = NULL,
use_cache = FALSE,
return_graph = FALSE
)
```

Arguments

- `pathway`: pathway ID to be passed to `pathway()`
- `set`: vector of identifiers, or named vector of numeric values
- `how`: if `all`, if node contains multiple IDs separated by `sep`, highlight if all the IDs are in query. if `any`, highlight if one of the IDs is in query.
- `num_combine`: combining function if multiple hits are obtained per node
- `name`: which column to search for
- `sep`: separator for node names
- `no_sep`: not separate node name
- `show_type`: entitie type, default to 'gene'
- `fill_color`: highlight color, default to 'tomato'
- `legend_name`: legend name, NULL to suppress
- `use_cache`: use cache or not
- `return_graph`: return tbl_graph instead of plot

Value

overlaid map

Examples

```r
highlight_entities("hsa04110", c("CDKN2A"), legend_name="interesting")
```

Description

identify if edges are involved in module reaction, and whether linked compounds are involved in the reaction. It would not be exactly the same as KEGG mapper. For instance, ‘R04293’ involved in ‘M00912’ is not included in KGML of ‘ko01100’.

Usage

```r
highlight_module(graph, kmo, name = "name", sep = " ", verbose = FALSE)
```
### highlight_set_edges

**Arguments**

- **graph**: tbl_graph
- **kmo**: kegg_module class object which stores reaction
- **name**: which column to search for
- **sep**: separator for node names
- **verbose**: show messages or not

**Value**

boolean vector

**Examples**

```r
## Highlight module within the pathway
graph <- create_test_pathway()
mo <- create_test_module()
graph <- graph |> highlight_module(mo)
```

### Description

identify if edges are involved in specific query. if multiple IDs are listed after separation by `sep`, only return TRUE if all the IDs are in the query.

**Usage**

```
highlight_set_edges(set, how = "all", name = "name", sep = " ", no_sep = FALSE)
```

**Arguments**

- **set**: set of identifiers
- **how**: if ‘all’, if node contains multiple IDs separated by `sep`, highlight if all the IDs are in query. if ‘any’, highlight if one of the IDs is in query.
- **name**: which column to search for
- **sep**: separator for node names
- **no_sep**: not separate node name

**Value**

boolean vector
Examples

```r
graph <- create_test_pathway()

## Specify edge column by `name`
## In this example, edges having `degradation` value in
## `subtype_name` column will be highlighted
graph <- graph |> activate("edges") |>
  mutate(hl=highlight_set_edges(c("degradation"), name="subtype_name"))
```

**Description**

identify if nodes are involved in specific query. if multiple IDs are listed after separation by `sep`, only return TRUE if all the IDs are in the query.

**Usage**

```r
highlight_set_nodes(set, how = "all", name = "name", sep = " ", no_sep = FALSE)
```

**Arguments**

- `set`: set of identifiers
- `how`: if `all`, if node contains multiple IDs separated by `sep`, highlight if all the IDs are in query. if `any`, highlight if one of the IDs is in query.
- `name`: which column to search for
- `sep`: separator for node names
- `no_sep`: not separate node name

**Value**

boolean vector

**Examples**

```r
graph <- create_test_pathway()

## Highlight set of nodes by specifying ID
graph <- graph |> mutate(hl=highlight_set_nodes(c("hsa:51428")))

## node column can be specified by `name` argument
graph <- graph |
  mutate(hl=highlight_set_nodes(c("DDX41"), name="graphics_name")))
```
**module**

module KEGG module parsing function

**Usage**

module(mid, use_cache = FALSE, directory = NULL)

**Arguments**

- `mid`: KEGG module ID
- `use_cache`: use cache
- `directory`: directory to save raw files

**Value**

list of module definition and reaction

**Examples**

```r
module("M00003")
```

---

**module_abundance**

module_abundance weighted mean abundance of fraction of present KO in the block

**Usage**

module_abundance(mod_id, vec, num = 1, calc = "weighted_mean")

**Arguments**

- `mod_id`: module ID
- `vec`: KO-named vector of abundance without prefix ‘ko:’
- `num`: definition number when multiple definitions are present
- `calc`: calculation of final results, mean or weighted_mean
Value
numeric value

Examples
module_abundance("M00003",c(1.2) |> setNames("K00927"))

Description
This converts module definitions consisting of KO identifiers to the expression by converting `+` and ` ` to `AND`, and `,` to `OR`. After that, KO IDs specified by `query` is inserted to expression by `TRUE` or `FALSE`, and is evaluated. Please feel free to contact the bug, or modules that cannot be calculated. (Module definitions consisting of module IDs [M*] cannot be calculated)

Usage
module_completeness(kmo, query, name = "1")

Arguments
- kmo: module object
- query: vector of KO
- name: name of definitions when multiple definitions are present

Details
Below is quoted from https://www.genome.jp/kegg/module.html
‘A space or a plus sign, representing a connection in the pathway or the molecular complex, is treated as an AND operator and a comma, used for alternatives, is treated as an OR operator. A minus sign designates an optional item in the complex.’

Value
tibble

Examples
## Assess completeness based on one KO input
test_complete <- module_completeness(create_test_module(), c("K00112"))
**module_text**

*module_text* Obtain textual representation of module definition for all the blocks

---

**Description**

module_text Obtain textual representation of module definition for all the blocks

**Usage**

```r
module_text(
  kmo,
  name = "1",
  candidate_ko = NULL,
  paint_colour = "tomato",
  convert = NULL
)
```

**Arguments**

- `kmo`: module object
- `name`: name of definition
- `candidate_ko`: KO to highlight
- `paint_colour`: color to highlight
- `convert`: named vector converting the KO to gene name

**Value**

textual description of module definitions

**Examples**

```r
mo <- create_test_module()
tex <- module_text(mo)
```

---

**multi_pathway_native**

*multi_pathway_native*

---

**Description**

If you want to combine multiple KEGG pathways with their native coordinates, supply this function a vector of pathway IDs and row number. This returns the joined graph or list of graphs in which the coordinates are altered to panel the pathways.
Usage

multi_pathway_native(pathways, row_num = 2, return_list = FALSE)

Arguments

- pathways: pathway vector
- row_num: row number
- return_list: return list of graphs instead of joined graph

Value

graph adjusted for the position

Examples

```r
## Pass multiple pathway IDs
multi_pathway_native(list("hsa04110","hsa03460"))
```

---

**network**

*KEGG network parsing function*

Description

parsing the network elements starting with N

Usage

network(nid, use_cache = FALSE, directory = NULL)

Arguments

- nid: KEGG NETWORK ID
- use_cache: use cache
- directory: directory to save raw files

Value

list of network definition

Examples

```r
network("N00002")
```
network_graph

Description

obtain tbl_graph of KEGG network

Usage

network_graph(kne, type = "definition")

Arguments

kne network object

Arguments

type definition or expanded

Value

tbl_graph

Examples

ne <- create_test_network()
neg <- network_graph(ne)

node_matrix

Description

given the matrix representing gene as row and sample as column, append the node value to node matrix and return tbl_graph object

Usage

node_matrix(
  graph,
  mat,
  gene_type = "SYMBOL",
  org = "hsa",
  org_db = org.Hs.eg.db,
  num_combine = mean
)

Arguments

- **num**
  - named vector or tibble with id and value column
- **num_combine**
  - how to combine number when multiple hit in the same node
- **name**
  - name of column to match for
- **how**
  - how to match the node IDs with the queries 'any' or 'all'

Value

- numeric vector
Examples

```r
graph <- create_test_pathway()
graph <- graph |
    mutate(num=node_numeric(c(1.1) |> setNames("hsa:6737")))
```

---

`obtain_sequential_module_definition`

**Description**

Given module definition and block number, Recursively obtain graphical representation of block and connect them by pseudo-nodes representing blocks.

**Usage**

```r
obtain_sequential_module_definition(kmo, name = "1", block = NULL)
```

**Arguments**

- **kmo**: module object
- **name**: name of definition when multiple definitions are present
- **block**: specify if need to parse specific block

**Value**

list of module definitions

**Examples**

```r
mo <- create_test_module()
sequential_mod <- obtain_sequential_module_definition(mo)
```

---

`output_overlay_image`

**Description**

The function first exports the image, combine it with the original image. Note that if the legend is outside the pathway image, the result will not show it correctly. Place the legend inside the panel by adding the theme such as `theme(legend.position=c(0.5, 0.5))`. 
output_overlay_image

Usage

output_overlay_image(
  gg,
  with_legend = TRUE,
  use_cache = TRUE,
  high_res = FALSE,
  res = 72,
  out = NULL,
  directory = NULL,
  transparent_colors = c("#FFFFFF", "#BFBFFF", "#BFFFBF", "#FF7F7F", "#808080"),
  unlink = TRUE,
  with_legend_image = FALSE,
  legend_horiz = FALSE,
  legend_space = 100
)

Arguments

  gg      ggraph object
  with_legend      if legend (group-box) is in gtable, output them
  use_cache      use BiocFileCache for caching the image
  high_res      use 2x resolution image
  res      resolution parameter passed to saving the ggplot2 image
  out      output file name
  directory      specify if you have already downloaded the image
  transparent_colors      transparent colors
  unlink      unlink the intermediate image
  with_legend_image      append legend image instead of using gtable
  legend_horiz      append legend to the bottom of the image
  legend_space      legend spacing specification (in pixel)

Details

If the legend must be placed outside the image, the users can set with_legend_image to TRUE. This will create another legend only image and concatenate it with the pathway image. legend_space option can be specified to control the spacing for the legend. If need to append horizontal legend, enable legend_horiz option.

By default, unlink option is enabled which means the function will delete the intermediate files.

Value

  output the image and return the path
Examples

```r
## Not run:
output_overlay_image(ggraph(pathway("hsa04110")))

## End(Not run)
```

Description

Overlay the raw KEGG pathway image on ggraph

Usage

```r
overlay_raw_map(
  pid = NULL,
  directory = NULL,
  transparent_colors = c("#FFFFFF", "#BFBFFF", "#BFFFBF"),
  adjust = FALSE,
  adjust_manual_x = NULL,
  adjust_manual_y = NULL,
  clip = FALSE,
  use_cache = TRUE,
  interpolate = TRUE,
  high_res = FALSE,
  fix_coordinates = TRUE
)
```

Arguments

- **pid**: pathway ID
- **directory**: directory to store images if not use cache
- **transparent_colors**: make these colors transparent to overlay Typical choice of colors would be: 
  "#CCCCCC", "#FFFFFF", "#BFBFFF", "#BFFFBF", "#7F7F7F", "#808080", "#ADADAD", "#838383", "#B3B3B3"
- **adjust**: adjust the x- and y-axis location by 0.5 in data coordinates
- **adjust_manual_x**: adjust the position manually for x-axis Override 'adjust'
- **adjust_manual_y**: adjust the position manually for y-axis Override 'adjust'
- **clip**: clip the both end of x- and y-axis by one dot
- **use_cache**: whether to use BiocFileCache()
interpolate parameter in annotation_raster()

high_res Use high resolution (2x) image for the overlay

fix_coordinates fix the coordinate (coord_fixed)

Value

ggplot2 object

Examples

```r
## Need `pathway_id` column in graph
## if the function is to automatically infer
graph <- create_test_pathway() |> mutate(pathway_id="hsa04110")
ggraph(graph) + overlay_raw_map()
```

Description

KEGG pathway parsing function

Usage

```r
pathway(
  pid,
  directory = NULL,
  use_cache = FALSE,
  group_rect_nudge = 2,
  node_rect_nudge = 0,
  invert_y = TRUE,
  add_pathway_id = TRUE,
  return_tbl_graph = TRUE,
  return_image = FALSE
)
```

Arguments

- `pid` pathway id
- `directory` directory to download KGML
- `use_cache` whether to use BiocFileCache
- `group_rect_nudge` nudge the position of group node default to add slight increase to show the group node
node_rect_nudge
  nudge the position of all node
invert_y
  invert the y position to match with R graphics
add_pathway_id
  add pathway id to graph, default to TRUE needed for the downstream analysis
return_tbl_graph
  return tbl_graph object, if FALSE, return igraph
return_image
  return the image URL

Value
  tbl_graph by default

Examples
  pathway("hsa04110")

pathway_abundance(pathway_abundance)

Description
  pathway_abundance

Usage
  pathway_abundance(id, vec, num = 1)

Arguments
  id
    pathway id
  vec
    named vector of abundance
  num
    number of module definition

Value
  numeric value

Examples
  pathway_abundance("ko00270", c(1.2) |>
    setNames("K00927"))
**pathway_info**

---

**Description**

obtain the list of pathway information

**Usage**

```r
pathway_info(pid, use_cache = FALSE, directory = NULL)
```

**Arguments**

- `pid`: KEGG Pathway id
- `use_cache`: whether to use cache
- `directory`: directory of file

**Value**

list of orthology and module contained in the pathway

**Examples**

```r
pathway_info("hsa04110")
```

---

**plot_kegg_network**

---

**Description**

plot the output of network_graph

**Usage**

```r
plot_kegg_network(g, layout = "nicely")
```

**Arguments**

- `g`: graph object returned by `network()`
- `layout`: layout to be used, default to nicely

**Value**

`ggplot2` object
Examples
ne <- create_test_network()
## Output of `network_graph` must be used with `plot_kegg_network`
neg <- network_graph(ne)
plt <- plot_kegg_network(neg)

plot_module_blocks

Description
wrapper function for plotting network representation of module definition blocks

Usage
plot_module_blocks(all_steps, layout = "kk")

Arguments
all_steps the result of `obtain_sequential_module_definition`
layout ggraph layout parameter

Value
ggplot2 object

Examples
mo <- create_test_module()
## The output of `obtain_sequential_module_definition`
## is used for `plot_module_blocks`
sequential_mod <- obtain_sequential_module_definition(mo)
plt <- plot_module_blocks(sequential_mod)

plot_module_text

Description
plot the text representation of KEGG modules

Usage
plot_module_text(plot_list, show_name = "name")
**process_line**

**Arguments**

- plot_list: the result of `module_text()`
- show_name: name column to be plotted

**Value**

ggplot2 object

**Examples**

```r
mo <- create_test_module()
## The output of `module_text` is used for `plot_module_text`

tex <- module_text(mo)
plt <- plot_module_text(tex)
```

**Description**

process the KGML containing graphics type of `line`, like global maps e.g. ko01100. Recursively add nodes and edges connecting them based on `coords` properties in KGML.

**Usage**

`process_line(g, invert_y = TRUE, verbose = FALSE)`

**Arguments**

- `g`: graph
- `invert_y`: whether to invert the position, default to TRUE should match with `pathway` function
- `verbose`: show progress

**Details**

We cannot show directed arrows, as coords are not ordered to show direction.

**Value**

tbl_graph
Examples

```r
## For those containing nodes with the graphic type of `line`,
## parse the coords attributes to edges.
gm_test <- create_test_pathway(line=TRUE)
test <- process_line(gm_test)
```

Description

process the kgml of global maps e.g. in ko01100

Usage

```r
process_reaction(g, single_edge = FALSE, keep_no_reaction = TRUE)
```

Arguments

- `g`: graph
- `single_edge`: discard one edge when edge type is 'reversible'
- `keep_no_reaction`: keep edges not related to reaction

Details

Typically, `process_line` function is used to draw relationships as in the original KGML positions, however, the `coords` properties is not considering the direction of reactions (substrate -> product), thus if it is preferred, `process_reaction` is used to populate new edges corresponding to 'substrate -> product' and 'product -> substrate' if the reaction is reversible.

Value

`tbl_graph`

Examples

```r
gm_test <- create_test_pathway(line=TRUE)
test <- process_reaction(gm_test)
```
Description

given enrichResult class object, return the ggplot object with raw KEGG map overlaid on enriched
pathway. Can be used with the function such as ‘clusterProfiler::enrichKEGG’ and ‘Microbiome-
Profiler::enrichKO()’

Usage

rawMap(
enrich,
pathway_number = 1,
pid = NULL,
fill_color = "red",
how = "any",
white_background = TRUE,
infer = FALSE
)

Arguments

enrich enrichResult or gseaResult class object, or list of them
pathway_number pathway number sorted by p-values
pid pathway id, override pathway_number if specified
fill_color color for genes
how how to match the node IDs with the queries ‘any’ or ‘all’
white_background fill background color white
infer if TRUE, append the prefix to queried IDs based on pathway ID

Value

ggraph with overlaid KEGG map

Examples

if (require("clusterProfiler")) {
cp <- enrichKEGG(c("1029","4171"))
## Multiple class object can be passed by list
rawMap(list(cp,cp), pid="hsa04110")
}
Description

given named vector of quantitative values, return the ggplot object with raw KEGG map overlaid. Colors can be changed afterwards.

Usage

rawValue(
  values,
  pid = NULL,
  column = "name",
  show_type = "gene",
  how = "any",
  white_background = TRUE,
  auto_add = FALSE,
  man_graph = NULL
)

Arguments

values named vector, or list of them
pid pathway id
column column name on node table of the graph
show_type type to be shown typically, "gene", "ortholog", or "compound"
how how to match the node IDs with the queries 'any' or 'all'
white_background fill background color white
auto_add automatically add prefix based on pathway prefix
man_graph provide manual tbl_graph

Value

ggraph with overlaid KEGG map

Examples

## Colorize by passing the named vector of numeric values
rv <- rawValue(c(1.1) |> setNames("hsa:6737"),
  man_graph=create_test_pathway())
return_line_compounds

Description

In the map, where lines are converted to edges, identify compounds that are linked by the reaction. Give the original edge ID of KGML (orig.id in edge table), and return the original compound node ID.

Usage

return_line_compounds(g, orig)

Arguments

g  tbl_graph object
orig  original edge ID

Value

vector of original compound node IDs

Examples

## For those containing nodes with the graphic type of `line`
## This returns no IDs as no edges are present
gm_test <- create_test_pathway(line=TRUE)
test <- process_line(gm_test) |> return_line_compounds(1)

stamp

Description

place stamp on the specified node

Usage

stamp(name, color = "red", which_column = "name", xval = 2, yval = 2)

Arguments

name  name of the nodes
color  color of the stamp
which_column  which node column to search
xval  adjustment value for x-axis
yval  adjustment value for y-axis
Value

ggplot2 object

Examples

test_pathway <- create_test_pathway()
plt <- ggraph(test_pathway, layout="manual", x=x, y=y) +
      stamp("hsa:6737")
Index

add_title, 3
append_cp, 4
append_label_position, 4
assign_deseq2, 5
carrow, 6
combine_with_bnlearn, 7
convert_id, 7
create_test_module, 9
create_test_network, 9
create_test_pathway, 10
dge_matrix, 10
dge_numeric, 11
dge_numeric_sum, 12
geom_kegg, 13
geom_node_rect, 13
geom_node_rect_kegg, 14
geom_node_shadowtext, 15
get_module_attribute, 16
get_module_attribute, kegg_module-method, 16
get_network_attribute, 17
get_network_attribute, kegg_network-method, 17
ggkegg, 18
ggkeggsave, 19
ggplot_add.geom_kegg, 20
ggplot_add.geom_node_rect_kegg, 20
ggplot_add.overlay_raw_map, 21
ggplot_add.stamp, 22
highlight_entities, 22
highlight_module, 23
highlight_set_edges, 24
highlight_set_nodes, 25
module, 26
module_abundance, 26
module_completeness, 27
module_text, 28
multi_pathway_native, 28
network, 29
network_graph, 30
node_matrix, 30
node_numeric, 31
obtain_sequential_module_definition, 32
output_overlay_image, 32
overlay_raw_map, 34
pathway, 35
pathway_abundance, 36
pathway_info, 37
plot_kegg_network, 37
plot_module_blocks, 38
plot_module_text, 38
process_line, 39
process_reaction, 40
rawMap, 41
rawValue, 42
return_line_compounds, 43
stamp, 43