

# Package ‘ggkegg’

April 25, 2024

**Type** Package

**Title** KEGG pathway visualization by ggplot2

**Version** 1.1.18

**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** devel

**git\_last\_commit** a79c20b

**git\_last\_commit\_date** 2024-04-18

**Repository** Bioconductor 3.19

**Date/Publication** 2024-04-25

**Author** Noriaki Sato [cre, aut]

**Maintainer** Noriaki Sato <nori@hgc.jp>

## Contents

add_title . . . . .	3
append_cp . . . . .	4
append_label_position . . . . .	4
assign_deseq2 . . . . .	5
carrow . . . . .	6
combine_with_blearn . . . . .	7
convert_id . . . . .	7
create_test_module . . . . .	9
create_test_network . . . . .	9
create_test_pathway . . . . .	10
edge_matrix . . . . .	10
edge_numeric . . . . .	11
edge_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

*add\_title* 3

<i>plot_kegg_network</i> . . . . .	37
<i>plot_module_blocks</i> . . . . .	38
<i>plot_module_text</i> . . . . .	38
<i>process_line</i> . . . . .	39
<i>process_reaction</i> . . . . .	40
<i>rawMap</i> . . . . .	41
<i>rawValue</i> . . . . .	42
<i>return_line_compounds</i> . . . . .	43
<i>stamp</i> . . . . .	43

**Index** 45

---

<i>add_title</i>	<i>addTitle</i>
------------------	-----------------

---

### 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

<i>out</i>	the image
<i>title</i>	the title
<i>size</i>	the size
<i>height</i>	title height
<i>color</i>	bg color
<i>titleColor</i>	title color
<i>gravity</i>	positioning of the title in the blank image

### Value

output the image

---

append_cp	<i>append_cp</i>
-----------	------------------

---

**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**

```
graph <- create_test_pathway()
nodes <- graph |> data.frame()
if (require("clusterProfiler")) {
  cp <- enrichKEGG(nodes$name |>
    strsplit(":", "|") |>
    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"))
}
```

---

append_label_position	<i>append_label_position</i>
-----------------------	------------------------------

---

**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.

**Usage**

```
append_label_position(g)
```

**Arguments**

g                    graph

**Value**

tbl\_graph

**Examples**

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

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

**Arguments**

res                    The result() of DESeq()

column                column of the numeric attribute, default to log2FoldChange

gene\_type             default to SYMBOL

org\_db                organism database to convert ID to ENTREZID

org                    organism ID in KEGG

numeric\_combine      how to combine multiple numeric values  
name                  column name for ID in tbl\_graph nodes

**Value**

numeric vector

**Examples**

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

---

carrow

*carrow*

---

**Description**

make closed type arrow

**Usage**

```
carrow(length = unit(2, "mm"))
```

**Arguments**

length                  arrow length in unit()

**Value**

arrow()

**Examples**

```
carrow()
```

---

combine\_with\_bnlearn    *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

pg	reference graph (output of ‘pathway’)
str	strength data.frame
av	averaged network to plot
prefix	add prefix to node name of original averaged network like, ‘hsa:’ or ‘ko:’.
how	‘any’ or ‘all’

### 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                    *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
edge	if converting edges

**Value**

vector containing converted IDs

**Examples**

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



---

`create_test_module`     *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`     *create\_test\_network*

---

**Description**

`create_test_network`

**Usage**

```
create_test_network()
```

**Value**

test network

**Examples**

```
create_test_network()
```

---

create\_test\_pathway     *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             *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

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

**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**

```
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")
```

---

edge_numeric	<i>edge_numeric</i>
--------------	---------------------

---

**Description**

add numeric attribute to edge of tbl\_graph

**Usage**

```
edge_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

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

---

edge_numeric_sum	<i>edge_numeric_sum</i>
------------------	-------------------------

---

## 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

```
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

```
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`*geom\_kegg*

---

### Description

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

### Usage

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

### Arguments

<code>edge_color</code>	color attribute to edge
<code>node_label</code>	column name for node label
<code>group_color</code>	border color for group node rectangles
<code>parallel</code>	use <code>geom_edge_parallel()</code> instead of <code>geom_edge_link()</code>

### Value

ggplot2 object

### Examples

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

---

`geom_node_rect`*geom\_node\_rect*

---

### Description

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

**Usage**

```
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**

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

---

geom\_node\_rect\_kegg    *geom\_node\_rect\_kegg*

---

**Description**

Wrapper function for plotting a certain type of nodes with background color with geom\_node\_rect()

**Usage**

```
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

## Examples

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

---

geom\_node\_shadowtext *geom\_node\_shadowtext*

---

## Description

Plot shadowtext at node position, use StatFilter in ggraph

## Usage

```
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

```
test_pathway <- create_test_pathway()
plt <- ggraph(test_pathway, layout="manual", x=x, y=y) +
  geom_node_shadowtext(aes(label=name))
```

---

`get_module_attribute`    *get\_module\_attribute*

---

**Description**

get slot from 'kegg\_module' class

**Usage**

```
get_module_attribute(x, attribute)
```

**Arguments**

<code>x</code>	kegg_module class object
<code>attribute</code>	pass to <code>get_module_attribute</code>

**Value**

attribute of kegg\_module

---

`get_module_attribute,kegg_module-method`  
*get\_module\_attribute*

---

**Description**

get the kegg\_module class attribute

**Usage**

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

**Arguments**

<code>x</code>	kegg_module class object
<code>attribute</code>	slot name

**Value**

attribute of kegg\_module



---

*get\_network\_attribute* *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`

---

`get_network_attribute`, kegg\_network-method  
*get\_network\_attribute*

---

**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`

ggkegg

*ggkegg***Description**

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

**Usage**

```
ggkegg(
  pid,
  layout = "native",
  return_igraph = FALSE,
  return_tbl_graph = FALSE,
  pathway_number = 1,
  convert_org = NULL,
  convert_first = TRUE,
  convert_collapse = 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
convert_collapse	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 object. If 'network', return 'tbl\_graph'.

**Value**

ggplot2 object

**Examples**

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

---

ggkeggsave

*ggkeggsave*

---

**Description**

save the image respecting the original width and height of the image. Only applicable for the ggplot object including 'overlay\_raw\_map' layers.

**Usage**

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

**Arguments**

<code>filename</code>	file name of the image
<code>plot</code>	plot to be saved
<code>dpi</code>	dpi, passed to ggsave
<code>wscale</code>	width scaling factor for pixel to inches
<code>hscale</code>	height scaling factor fo pixel to inches

**Value**

save the image

---

`ggplot_add.geom_kegg` *ggplot\_add.geom\_kegg*

---

**Description**

`ggplot_add.geom_kegg`

**Usage**

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

**Arguments**

<code>object</code>	An object to add to the plot
<code>plot</code>	The ggplot object to add object to
<code>object_name</code>	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()
```

---

`ggplot_add.geom_node_rect_kegg`  
*ggplot\_add.geom\_node\_rect\_kegg*

---

**Description**

`ggplot_add.geom_node_rect_kegg`

**Usage**

```
## S3 method for class 'geom_node_rect_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()
plt <- ggraph(test_pathway, layout="manual", x=x, y=y) +
  geom_node_rect_kegg(type="gene")
```

---

ggplot\_add.overlay\_raw\_map

*ggplot\_add.overlay\_raw\_map*

---

**Description**

ggplot\_add.overlay\_raw\_map

**Usage**

```
## S3 method for class 'overlay_raw_map'
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**

```
## 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()
```

---

ggplot_add.stamp	<i>ggplot_add.stamp</i>
------------------	-------------------------

---

### Description

ggplot\_add.stamp

### Usage

```
## S3 method for class 'stamp'  
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()  
plt <- ggraph(test_pathway, layout="manual", x=x, y=y) +  
  stamp("hsa:6737")
```

---

highlight_entities	<i>highlight_entities</i>
--------------------	---------------------------

---

### 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 = ",",
```

```

    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

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

---

highlight_module	<i>highlight_module</i>
------------------	-------------------------

---

### 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

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

**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**

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

---

`highlight_set_edges`     *highlight\_set\_edges*

---

**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

```
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"))
```

---

highlight\_set\_nodes    *highlight\_set\_nodes*

---

## 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

```
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

```
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	<i>module KEGG module parsing function</i>
--------	--------------------------------------------

---

**Description**

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**

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

---

module_abundance	<i>module_abundance weighted mean abundance of fraction of present KO in the block</i>
------------------	----------------------------------------------------------------------------------------

---

**Description**

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"))
```

---

module\_completeness    *module\_completeness*

---

**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	<i>module_text</i> Obtain textual representation of module definition for all the blocks
-------------	------------------------------------------------------------------------------------------

---

### Description

module\_text Obtain textual representation of module definition for all the blocks

### Usage

```
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

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

---

multi_pathway_native	<i>multi_pathway_native</i>
----------------------	-----------------------------

---

### 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**

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

---

network	<i>KEGG network parsing function</i>
---------	--------------------------------------

---

**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**

```
network("N00002")
```

---

network_graph	<i>network_graph</i>
---------------	----------------------

---

**Description**

obtain tbl\_graph of KEGG network

**Usage**

```
network_graph(kne, type = "definition")
```

**Arguments**

kne	network object
type	definition or expanded

**Value**

tbl\_graph

**Examples**

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

---

node_matrix	<i>node_matrix</i>
-------------	--------------------

---

**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**

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**

```
## Append data.frame to tbl_graph
graph <- create_test_pathway()
num_df <- data.frame(row.names=c("6737","51428"),
                    "sample1"=c(1.1,1.2),
                    "sample2"=c(1.5,2.2),
                    check.names=FALSE)
graph <- graph |> node_matrix(num_df, gene_type="ENTREZID")
```

---

node\_numeric

*node\_numeric*

---

**Description**

simply add numeric attribute to node of tbl\_graph

**Usage**

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

**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**

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

---

```
obtain_sequential_module_definition
  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**

```
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**

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

---

```
output_overlay_image  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)).



**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", "#7F7F7F", "#808080"),
  unlink = TRUE,
  with_legend_image = FALSE,
  legend_horiz = FALSE,
  legend_space = 100
)
```

**Arguments**

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

```
## Not run:
  ouput_overlay_image(ggraph(pathway("hsa04110")))

## End(Not run)
```

---

overlay_raw_map	<i>overlay_raw_map</i>
-----------------	------------------------

---

**Description**

Overlay the raw KEGG pathway image on ggraph

**Usage**

```
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", "#7F7F7F"
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**

```
## 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()
```

---

pathway	<i>pathway</i>
---------	----------------

---

**Description**

KEGG pathway parsing function

**Usage**

```
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	<i>pathway_abundance</i>
-------------------	--------------------------

---

**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	<i>pathway_info</i>
--------------	---------------------

---

**Description**

obtain the list of pathway information

**Usage**

```
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**

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

---

plot_kegg_network	<i>plot_kegg_network</i>
-------------------	--------------------------

---

**Description**

plot the output of network\_graph

**Usage**

```
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      *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      *plot\_module\_text*

---

**Description**

plot the text representation of KEGG modules

**Usage**

```
plot_module_text(plot_list, show_name = "name")
```

**Arguments**

plot_list	the result of 'module_text()'
show_name	name column to be plotted

**Value**

ggplot2 object

**Examples**

```
mo <- create_test_module()

## The output of `module_text` is used for `plot_module_text`
tex <- module_text(mo)
plt <- plot_module_text(tex)
```

---

process_line	<i>process_line</i>
--------------	---------------------

---

**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

```
## 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)
```

---

process_reaction	<i>process_reaction</i>
------------------	-------------------------

---

## Description

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

## Usage

```
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

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



---

rawMap	<i>rawMap</i>
--------	---------------

---

## 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")  
}
```

---

rawValue	<i>rawValue</i>
----------	-----------------

---

### 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 *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 *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  
  
edge\_matrix, 10  
edge\_numeric, 11  
edge\_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