

# Package ‘ReporterScore’

February 20, 2026

**Type** Package

**Title** Generalized Reporter Score-Based Enrichment Analysis for Omics Data

**Version** 0.2.5

**Description** Inspired by the classic 'RSA', we developed the improved 'Generalized Reporter Score-based Analysis (GRSA)' method, implemented in the R package 'ReporterScore', along with comprehensive visualization methods and pathway databases. 'GRSA' is a threshold-free method that works well with all types of biomedical features, such as genes, chemical compounds, and microbial species. Importantly, the 'GRSA' supports multi-group and longitudinal experimental designs, because of the included multi-group-compatible statistical methods.

**License** GPL-3

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**Imports** magrittr, dplyr, stats, ggplot2 (>= 3.2.0), pcutils (>= 0.2.5), utils, scales, ggnewscale, ggrepel, reshape2, stringr, foreach

**Suggests** knitr, rmarkdown, plyr, e1071, factoextra, snow, doSNOW, pheatmap, readr, R.utils, KEGGREST, ggkegg, clusterProfiler, enrichplot, pathview, GSA, vegan, MetaNet, igraph, ggraph, PADOG, safe, rSEA, GSVA

**Depends** R (>= 4.2.0)

**VignetteBuilder** knitr

**BugReports** <https://github.com/Asa12138/ReporterScore/issues>

**URL** <https://github.com/Asa12138/ReporterScore>

**NeedsCompilation** no

**Author** Chen Peng [aut, cre] (ORCID: <<https://orcid.org/0000-0002-9449-7606>>)

**Maintainer** Chen Peng <bfzede@gmail.com>

**Repository** CRAN

**Date/Publication** 2026-02-20 14:50:14 UTC

## Contents

cm_test_k . . . . .	3
combine_rs_res . . . . .	4
Compound_htable . . . . .	5
CPDlist . . . . .	5
custom_modulelist . . . . .	6
custom_modulelist_from_org . . . . .	7
c_net_from_pathway_xml . . . . .	8
export_report_table . . . . .	8
gene2ko . . . . .	9
genedf . . . . .	9
get_all_pathway_net_index . . . . .	10
get_features . . . . .	10
get_reporter_score . . . . .	11
GOList . . . . .	12
hsa_kegg_pathway . . . . .	13
ko.test . . . . .	13
KOList . . . . .	14
KO_abundance . . . . .	15
KO_enrich . . . . .	15
KO_fisher . . . . .	16
KO_gsa . . . . .	17
KO_gsea . . . . .	18
KO_gsva . . . . .	20
KO_htable . . . . .	20
KO_padog . . . . .	21
KO_safe . . . . .	22
KO_sea . . . . .	23
load_CARDinfo . . . . .	23
load_CAZy_info . . . . .	24
load_Enzyme_info . . . . .	24
load_GOList . . . . .	25
load_htable . . . . .	25
load_pathway_xml_ls . . . . .	27
mmu_kegg_pathway . . . . .	27
modify_description . . . . .	28
Module_htable . . . . .	28
parse_enzyme_dat . . . . .	29
Pathway_htable . . . . .	29
pathway_net_index . . . . .	29
plot.cm_res . . . . .	30
plot_enrich_res . . . . .	31
plot_features_box . . . . .	32
plot_features_distribution . . . . .	33
plot_features_heatmap . . . . .	34
plot_features_in_pathway . . . . .	35
plot_features_network . . . . .	36

plot_htable . . . . .	38
plot_KEGG_map . . . . .	38
plot_pathway_net . . . . .	40
plot_report . . . . .	40
plot_report_circle_packing . . . . .	42
plot_significance . . . . .	43
print.reporter_score . . . . .	43
print.rs_by_cm . . . . .	44
pvalue2zs . . . . .	44
reporter_score . . . . .	46
reporter_score_res . . . . .	49
RSA_by_cm . . . . .	50
update_CARDinfo . . . . .	51
update_CAzy_info . . . . .	52
update_Enzyme_info . . . . .	53
update_GOlist . . . . .	53
update_KEGG . . . . .	54
update_pathway_xml_ls . . . . .	55
up_level_KO . . . . .	55

**Index****57**


---

cm_test_k	<i>Test the proper clusters k for c_means</i>
-----------	---

---

**Description**

Test the proper clusters k for c\_means

C-means cluster

**Usage**

```
cm_test_k(otu_group, filter_var, fast = TRUE)
```

```
c_means(otu_group, k_num, filter_var)
```

**Arguments**

otu_group	standardize data
filter_var	filter the highest var
fast	whether do the gap_stat?
k_num	cluster number

**Value**

ggplot  
ggplot

**See Also**

Other C\_means: [RSA\\_by\\_cm\(\)](#)

**Examples**

```
if (requireNamespace("e1071") && requireNamespace("factoextra")) {
  data(otutab, package = "pcutils")
  pcutils::hebing(otutab, metadata$Group) -> otu_group
  cm_test_k(otu_group, filter_var = 0.7)
  cm_res <- c_means(otu_group, k_num = 3, filter_var = 0.7)
  plot(cm_res, 0.8)
}
```

---

combine_rs_res	<i>Combine the results of 'step by step GRSA'</i>
----------------	---

---

**Description**

Combine the results of 'step by step GRSA'

**Usage**

```
combine_rs_res(kodf, group, metadata, ko_stat, reporter_s, modulelist = NULL)
```

**Arguments**

kodf	KO_abundance table, rowname are feature ids (e.g. K00001 if feature="ko"; PEX11A if feature="gene"; C00024 if feature="compound"), colnames are samples.
group	The comparison groups (at least two categories) in your data, one column name of metadata when metadata exist or a vector whose length equal to columns number of kodf. And you can use factor levels to change order.
metadata	sample information data.frame contains group
ko_stat	result of <a href="#">pvalue2zs</a>
reporter_s	result of <a href="#">get_reporter_score</a>
modulelist	NULL or customized modulelist dataframe, must contain 'id', 'K_num', 'KOs', 'Description' columns. Take the 'KOlist' as example, use <a href="#">custom_modulelist</a> .

**Value**

reporter\_score object

**See Also**

Other GRSA: [get\\_reporter\\_score\(\)](#), [ko.test\(\)](#), [pvalue2zs\(\)](#), [reporter\\_score\(\)](#)

**Examples**

```

data("KO_abundance_test")
ko_pvalue <- ko.test(KO_abundance, "Group", metadata)
ko_stat <- pvalue2zs(ko_pvalue, mode = "directed")
reporter_s1 <- get_reporter_score(ko_stat, perm = 499)
reporter_res <- combine_rs_res(KO_abundance, "Group", metadata, ko_stat, reporter_s1)

```

---

Compound_htable	<i>Compound htable from 'KEGG'</i>
-----------------	------------------------------------

---

**Description**

Compound htable from 'KEGG'

**See Also**

Other data: [CPDlist](#), [Golist](#), [KO\\_htable](#), [Kolist](#), [Module\\_htable](#), [Pathway\\_htable](#), [hsa\\_kegg\\_pathway](#), [mmu\\_kegg\\_pathway](#)

---

CPDlist	<i>The CPDlist used for enrichment.</i>
---------	---

---

**Description**

an list contains two data.frame named pathway and module.

**Format**

four columns in each data.frame.

**id** "map0010" or "M00001"

**K\_num** contains how many Compounds in this pathway or module

**KOs** Compounds name

**Description** the description of this pathway or module

**See Also**

Other data: [Compound\\_htable](#), [Golist](#), [KO\\_htable](#), [Kolist](#), [Module\\_htable](#), [Pathway\\_htable](#), [hsa\\_kegg\\_pathway](#), [mmu\\_kegg\\_pathway](#)

---

custom\_modulelist      *Build a custom modulelist*

---

### Description

Build a custom modulelist

Transform a modulelist to a list

### Usage

```
custom_modulelist(pathway2ko, pathway2desc = NULL, verbose = TRUE)
```

```
transform_modulelist(mymodulelist, mode = 1)
```

### Arguments

pathway2ko      user input annotation of Pathway to KO mapping, a data.frame of 2 column with pathway and ko.

pathway2desc    user input of Pathway TO Description mapping, a data.frame of 2 column with pathway and description.

verbose          verbose

mymodulelist    mymodulelist

mode            1~2

### Value

a custom modulelist

modulelist

### See Also

Other modulelist: [custom\\_modulelist\\_from\\_org\(\)](#), [get\\_features\(\)](#)

Other modulelist: [custom\\_modulelist\\_from\\_org\(\)](#), [get\\_features\(\)](#)

### Examples

```
mydat <- data.frame(pathway = paste0("PATHWAY", rep(seq_len(2), each = 5)), ko = paste0("K", 1:10))
mymodulelist <- custom_modulelist(mydat)
print(mymodulelist)
transform_modulelist(mymodulelist)
```

---

`custom_modulelist_from_org`*Custom modulelist from a specific organism*

---

## Description

Custom modulelist from a specific organism

## Usage

```
custom_modulelist_from_org(  
  org = "hsa",  
  feature = "ko",  
  gene = "symbol",  
  verbose = TRUE  
)
```

## Arguments

<code>org</code>	kegg organism, listed in <a href="https://www.genome.jp/kegg/catalog/org_list.html">https://www.genome.jp/kegg/catalog/org_list.html</a> , default, "hsa"
<code>feature</code>	one of "ko", "gene", "compound"
<code>gene</code>	one of "symbol", "id"
<code>verbose</code>	logical

## Value

modulelist

## See Also

Other modulelist: [custom\\_modulelist\(\)](#), [get\\_features\(\)](#)

## Examples

```
hsa_pathway <- custom_modulelist_from_org(org = "hsa", feature = "gene")
```

---

c\_net\_from\_pathway\_xml

*Create a network from KEGG pathway XML files*

---

**Description**

Create a network from KEGG pathway XML files

**Usage**

```
c_net_from_pathway_xml(pathway_xml)
```

**Arguments**

pathway\_xml     A 'tbl\_graph' or 'igraph' object, or a file path to a KEGG XML file.

**Value**

A 'metanet' object representing the pathway network.

**See Also**

plot\_pathway\_net

---

export\_report\_table

*Export report score result tables*

---

**Description**

Export report score result tables

**Usage**

```
export_report_table(reporter_res, dir_name, overwrite = FALSE)
```

**Arguments**

reporter\_res     a reporter\_score object or rs\_by\_cm object  
dir\_name         the directory to save the report tables  
overwrite        overwrite the existed files or not, default is FALSE.

**Value**

No return value

---

gene2ko                      *Transfer gene symbol table to KO table*

---

### Description

You can use 'clusterProfiler::bitr()' to transfer your table from other gene\_id to gene\_symbol.

### Usage

```
gene2ko(genedf, org = "hsa")
```

### Arguments

genedf                      ,rowname is gene symbol (e.g. PFKM), colnames is samples  
 org                         kegg organism, listed in 'https://www.genome.jp/kegg/catalog/org\_list.html', default, 'hsa'

### Value

kodf

### Examples

```
data("genedf")
K0df <- gene2ko(genedf, org = "hsa")
```

---

genedf                      *human gene table*

---

### Description

human gene table

### See Also

Other test\_data: [KO\\_abundance](#), [reporter\\_score\\_res](#)

---

```
get_all_pathway_net_index
```

*Create an index for all KEGG pathway networks*

---

### Description

Create an index for all KEGG pathway networks

### Usage

```
get_all_pathway_net_index(pathway_xml_ls = NULL, org = NULL)
```

### Arguments

`pathway_xml_ls` A list of KEGG pathway XML files, where each element is a 'tbl\_graph' or 'igraph' object.

`org` Character, the KEGG organism code (e.g., "hsa" for human). If 'NULL', uses "ko" as the default prefix for pathway IDs.

### Value

A data frame containing indices for all pathways, including pathway IDs and their attributes.

---

```
get_features
```

*get features in a modulelist*

---

### Description

get features in a modulelist

### Usage

```
get_features(map_id = "map00010", ko_stat = NULL, modulelist = NULL)
```

### Arguments

`map_id` `map_id` in modulelist

`ko_stat` NULL or `ko_stat` result from [pvalue2zs](#)

`modulelist` NULL or customized modulelist dataframe, must contain 'id', 'K\_num', 'KOs', 'Description' columns. Take the 'Kolist' as example, use [custom\\_modulelist](#).

### Value

KOids, or data.frame with these KOids.

**See Also**

Other modulelist: [custom\\_modulelist\(\)](#), [custom\\_modulelist\\_from\\_org\(\)](#)

**Examples**

```
get_features(map_id = "map00010")
```

---

get_reporter_score	<i>Calculate reporter score</i>
--------------------	---------------------------------

---

**Description**

Calculate reporter score

**Usage**

```
get_reporter_score(
  ko_stat,
  type = c("pathway", "module")[1],
  feature = "ko",
  threads = 1,
  modulelist = NULL,
  perm = 4999,
  verbose = TRUE,
  p.adjust.method2 = "BH",
  min_exist_KO = 3,
  max_exist_KO = 600
)
```

**Arguments**

ko_stat	ko_stat result from <a href="#">pvalue2zs</a>
type	'pathway' or 'module' for default KOList for microbiome, 'CC', 'MF', 'BP', 'ALL' for default GOList for homo sapiens. And org in listed in 'https://www.genome.jp/kegg/catalog/org' such as 'hsa' (if your kodf is come from a specific organism, you should specify type here).
feature	one of 'ko', 'gene', 'compound'
threads	default 1
modulelist	NULL or customized modulelist dataframe, must contain 'id', 'K_num', 'KOs', 'Description' columns. Take the 'KOList' as example, use <a href="#">custom_modulelist</a> .
perm	permutation number, default: 4999.
verbose	logical
p.adjust.method2	p.adjust.method for the correction of ReporterScore, see <a href="#">p.adjust</a>

**min\_exist\_KO** min exist KO number in a pathway (default, 3, when a pathway contains KOs less than 3, there will be no RS)

**max\_exist\_KO** max exist KO number in a pathway (default, 600, when a pathway contains KOs more than 600, there will be no RS)

**Value**

reporter\_res data.frame

**See Also**

Other GRSA: [combine\\_rs\\_res\(\)](#), [ko.test\(\)](#), [pvalue2zs\(\)](#), [reporter\\_score\(\)](#)

**Examples**

```
data("KO_abundance_test")
ko_pvalue <- ko.test(KO_abundance, "Group", metadata)
ko_stat <- pvalue2zs(ko_pvalue, mode = "directed")
reporter_s1 <- get_reporter_score(ko_stat, perm = 499)
```

---

GOlist

*The GOlist used for enrichment.*

---

**Description**

an list contains three data.frame named BP, CC, MF.

**Format**

four columns in each data.frame.

**id** "map0010" or "M00001"

**K\_num** contains how many Genes in this GO term

**KOs** Genes name

**Description** the description of this GO term

**See Also**

Other data: [CPDlist](#), [Compound\\_htable](#), [KO\\_htable](#), [Kolist](#), [Module\\_htable](#), [Pathway\\_htable](#), [hsa\\_kegg\\_pathway](#), [mmu\\_kegg\\_pathway](#)

---

hsa_kegg_pathway	<i>pathway information for "hsa"</i>
------------------	--------------------------------------

---

**Description**

pathway information for "hsa"

**See Also**

Other data: [CPDlist](#), [Compound\\_htable](#), [Golist](#), [KO\\_htable](#), [Kolist](#), [Module\\_htable](#), [Pathway\\_htable](#), [mmu\\_kegg\\_pathway](#)

---

ko.test	<i>Differential analysis or Correlation analysis for KO-abundance table</i>
---------	---

---

**Description**

Differential analysis or Correlation analysis for KO-abundance table

**Usage**

```
ko.test(
  kofd,
  group,
  metadata = NULL,
  method = "wilcox.test",
  pattern = NULL,
  p.adjust.method1 = "none",
  threads = 1,
  verbose = TRUE
)
```

**Arguments**

kofd	KO_abundance table, rowname are feature ids (e.g. K00001 if feature="ko"; PEX11A if feature="gene"; C00024 if feature="compound"), colnames are samples.
group	The comparison groups (at least two categories) in your data, one column name of metadata when metadata exist or a vector whose length equal to columns number of kofd. And you can use factor levels to change order.
metadata	sample information data.frame contains group
method	the type of test. Default is 'wilcox.test'. Allowed values include: <ul style="list-style-type: none"> <li>• <a href="#">t.test</a> (parametric) and <a href="#">wilcox.test</a> (non-parametric). Perform comparison between two groups of samples. If the grouping variable contains more than two levels, then a pairwise comparison is performed.</li> </ul>

- [anova](#) (parametric) and [kruskal.test](#) (non-parametric). Perform one-way ANOVA test comparing multiple groups.
- 'pearson', 'kendall', or 'spearman' (correlation), see [cor](#).

**pattern** a named vector matching the group, e.g. `c('G1'=1,'G2'=3,'G3'=2)`, use the correlation analysis with specific pattern to calculate p-value.

**p.adjust.method1** p.adjust.method for 'ko.test', see [p.adjust](#)

**threads** default 1

**verbose** logical

### Value

ko\_pvalue data.frame

### See Also

Other GRSA: [combine\\_rs\\_res\(\)](#), [get\\_reporter\\_score\(\)](#), [pvalue2zs\(\)](#), [reporter\\_score\(\)](#)

### Examples

```
data("KO_abundance_test")
ko_pvalue <- ko.test(KO_abundance, "Group", metadata)
```

---

KOlis

*The KOlis used for enrichment.*

---

### Description

an list contains two data.frame named pathway and module.

### Format

four columns in each data.frame.

**id** "map0010" or "M00001"

**K\_num** contains how many KOs in this pathway or module

**KOs** KOs name

**Description** the description of this pathway or module

### See Also

Other data: [CPDlist](#), [Compound\\_htable](#), [G0list](#), [KO\\_htable](#), [Module\\_htable](#), [Pathway\\_htable](#), [hsa\\_kegg\\_pathway](#), [mmu\\_kegg\\_pathway](#)

---

KO_abundance	<i>The KOs abundance table and group table.</i>
--------------	---

---

**Description**

The KOs abundance table and group table.

The KOs abundance table and group table.

**See Also**

Other test\_data: [genedf](#), [reporter\\_score\\_res](#)

---

KO_enrich	<i>Perform enrichment analysis</i>
-----------	------------------------------------

---

**Description**

This function performs KO enrichment analysis using the ‘clusterProfiler’ package.

**Usage**

```
KO_enrich(
  ko_stat,
  padj_threshold = 0.05,
  logFC_threshold = NULL,
  add_mini = NULL,
  p.adjust.method = "BH",
  type = c("pathway", "module")[1],
  feature = "ko",
  modulelist = NULL,
  verbose = TRUE
)
```

```
as.enrich_res(gsea_res)
```

**Arguments**

ko\_stat ko\_stat dataframe from [ko.test](#).

padj\_threshold p.adjust threshold to determine whether a feature significant or not. p.adjust < padj\_threshold, default: 0.05

logFC\_threshold logFC threshold to determine whether a feature significant or not. abs(logFC)>logFC\_threshold, default: NULL

add\_mini add\_mini when calculate the logFC. e.g (10+0.1)/(0+0.1), default 0.05\*min(avg\_abundance)

p.adjust.method	The method used for p-value adjustment (default: "BH").
type	"pathway" or "module" for default Kolist_file.
feature	one of "ko", "gene", "compound"
modulelist	NULL or customized modulelist dataframe, must contain "id", "K_num", "KOs", "Description" columns. Take the 'Kolist' as example, use <a href="#">custom_modulelist</a> .
verbose	logical
gsea_res	gsea_res from KO_gsea

**Value**

A data frame containing the enrichment results.

enrich\_res object

**See Also**

Other common\_enrich: [KO\\_fisher\(\)](#), [KO\\_gsa\(\)](#), [KO\\_gsea\(\)](#), [KO\\_gsva\(\)](#), [KO\\_padog\(\)](#), [KO\\_safe\(\)](#), [KO\\_sea\(\)](#), [plot\\_enrich\\_res\(\)](#)

---

KO\_fisher

*Perform fisher's exact enrichment analysis*

---

**Description**

Perform fisher's exact enrichment analysis

**Usage**

```
KO_fisher(
  ko_stat,
  padj_threshold = 0.05,
  logFC_threshold = NULL,
  add_mini = NULL,
  p.adjust.method = "BH",
  type = c("pathway", "module")[1],
  feature = "ko",
  modulelist = NULL,
  verbose = TRUE
)
```

**Arguments**

ko_stat	ko_stat dataframe from <a href="#">ko.test</a> .
padj_threshold	p.adjust threshold to determine whether a feature significant or not. p.adjust < padj_threshold, default: 0.05
logFC_threshold	logFC threshold to determine whether a feature significant or not. abs(logFC)>logFC_threshold, default: NULL
add_mini	add_mini when calculate the logFC. e.g (10+0.1)/(0+0.1), default 0.05*min(avg_abundance)
p.adjust.method	The method used for p-value adjustment (default: "BH").
type	"pathway" or "module" for default Kolist_file.
feature	one of "ko", "gene", "compound"
modulelist	NULL or customized modulelist dataframe, must contain "id", "K_num", "KOs", "Description" columns. Take the 'Kolist' as example, use <a href="#">custom_modulelist</a> .
verbose	logical

**Value**

data.frame

**See Also**

Other common\_enrich: [KO\\_enrich\(\)](#), [KO\\_gsa\(\)](#), [KO\\_gsea\(\)](#), [KO\\_gsva\(\)](#), [KO\\_padog\(\)](#), [KO\\_safe\(\)](#), [KO\\_sea\(\)](#), [plot\\_enrich\\_res\(\)](#)

**Examples**

```
## use `fisher.test` from the `stats` package.
data("reporter_score_res")
fisher_res <- KO_fisher(reporter_score_res)
```

---

KO\_gsa *Perform gene set analysis*

---

**Description**

Perform gene set analysis

**Usage**

```
KO_gsa(
  reporter_res,
  method = "Two class unpaired",
  p.adjust.method = "BH",
  verbose = TRUE,
  perm = 1000,
  ...
)
```

**Arguments**

reporter_res	reporter_res
method	Problem type: "quantitative" for a continuous parameter; "Two class unpaired" ; "Survival" for censored survival outcome; "Multiclass" : more than 2 groups, coded 1,2,3...; "Two class paired" for paired outcomes, coded -1,1 (first pair), -2,2 (second pair), etc
p.adjust.method	"BH"
verbose	TRUE
perm	1000
...	additional parameters to <a href="#">GSA</a>

**Value**

enrich\_res object

**See Also**

Other common\_enrich: [KO\\_enrich\(\)](#), [KO\\_fisher\(\)](#), [KO\\_gsea\(\)](#), [KO\\_gsva\(\)](#), [KO\\_padog\(\)](#), [KO\\_safe\(\)](#), [KO\\_sea\(\)](#), [plot\\_enrich\\_res\(\)](#)

**Examples**

```
## use `GSA` from the `GSA` package.
if (requireNamespace("GSA")) {
  data("reporter_score_res")
  gsa_res <- KO_gsa(reporter_score_res, p.adjust.method = "none", perm = 200)
  plot(gsa_res)
}
```

---

KO\_gsea

*Perform gene set enrichment analysis*

---

**Description**

Perform gene set enrichment analysis

**Usage**

```
KO_gsea(
  ko_stat,
  weight = "logFC",
  add_mini = NULL,
  p.adjust.method = "BH",
  type = c("pathway", "module")[1],
```

```

    feature = "ko",
    modulelist = NULL,
    verbose = TRUE
  )

```

### Arguments

ko_stat	ko_stat dataframe from <a href="#">ko.test</a> .
weight	the metric used for ranking, default: logFC
add_mini	add_mini when calculate the logFC. e.g (10+0.1)/(0+0.1), default 0.05*min(avg_abundance)
p.adjust.method	The method used for p-value adjustment (default: "BH").
type	"pathway" or "module" for default Kolist_file.
feature	one of "ko", "gene", "compound"
modulelist	NULL or customized modulelist dataframe, must contain "id", "K_num", "KOs", "Description" columns. Take the 'Kolist' as example, use <a href="#">custom_modulelist</a> .
verbose	logical

### Value

DOSE object

### See Also

Other common\_enrich: [KO\\_enrich\(\)](#), [KO\\_fisher\(\)](#), [KO\\_gsa\(\)](#), [KO\\_gsva\(\)](#), [KO\\_padog\(\)](#), [KO\\_safe\(\)](#), [KO\\_sea\(\)](#), [plot\\_enrich\\_res\(\)](#)

### Examples

```

message("The following example require some time to run:")

## use `GSEA` from the `clusterProfiler` package.
if (requireNamespace("clusterProfiler")) {
  data("reporter_score_res")
  gsea_res <- KO_gsea(reporter_score_res, p.adjust.method = "none")
  enrichplot::gseaplot(gsea_res, geneSetID = data.frame(gsea_res)$ID[1])
  gsea_res_df <- as.enrich_res(gsea_res)
  plot(gsea_res_df)
}

```

---

KO_gsva	<i>Perform Gene Set Variation Analysis</i>
---------	--

---

**Description**

Perform Gene Set Variation Analysis

**Usage**

```
KO_gsva(
  reporter_res,
  verbose = TRUE,
  method = "wilcox.test",
  p.adjust.method = "BH",
  ...
)
```

**Arguments**

reporter_res	reporter_res
verbose	verbose
method	see <a href="#">ko.test</a>
p.adjust.method	p.adjust.method
...	additional parameters to <a href="#">gsva</a>

**Value**

enrich\_res

**See Also**

Other common\_enrich: [KO\\_enrich\(\)](#), [KO\\_fisher\(\)](#), [KO\\_gsa\(\)](#), [KO\\_gsea\(\)](#), [KO\\_padog\(\)](#), [KO\\_safe\(\)](#), [KO\\_sea\(\)](#), [plot\\_enrich\\_res\(\)](#)

---

KO_htable	<i>KO htable from 'KEGG'</i>
-----------	------------------------------

---

**Description**

KO htable from 'KEGG'

**See Also**

Other data: [CPDlist](#), [Compound\\_htable](#), [GOList](#), [KOlist](#), [Module\\_htable](#), [Pathway\\_htable](#), [hsa\\_kegg\\_pathway](#), [mmu\\_kegg\\_pathway](#)

---

KO_padog	<i>Perform Pathway Analysis with Down-weighting of Overlapping Genes (PADOG)</i>
----------	--

---

### Description

Perform Pathway Analysis with Down-weighting of Overlapping Genes (PADOG)

### Usage

```
KO_padog(  
  reporter_res,  
  verbose = TRUE,  
  perm = 1000,  
  p.adjust.method = "BH",  
  ...  
)
```

### Arguments

reporter_res	The input reporter result.
verbose	If TRUE, print verbose messages. Default is TRUE.
perm	The number of permutations. Default is 1000.
p.adjust.method	Method for p-value adjustment. Default is "BH".
...	Additional parameters to be passed to <a href="#">padog</a> function.

### Value

A data frame containing PADOG results for KO enrichment.

A data frame with columns "ID," "Description," "K\_num," "Exist\_K\_num," "p.value," and "p.adjust."

### See Also

Other common\_enrich: [KO\\_enrich\(\)](#), [KO\\_fisher\(\)](#), [KO\\_gsa\(\)](#), [KO\\_gsea\(\)](#), [KO\\_gsva\(\)](#), [KO\\_safe\(\)](#), [KO\\_sea\(\)](#), [plot\\_enrich\\_res\(\)](#)

---

`KO_safe`*Perform Significance Analysis of Function and Expression*

---

**Description**

Perform Significance Analysis of Function and Expression

**Usage**

```
KO_safe(  
  reporter_res,  
  verbose = TRUE,  
  perm = 1000,  
  C.matrix = NULL,  
  p.adjust.method = "BH",  
  ...  
)
```

**Arguments**

<code>reporter_res</code>	The input reporter result.
<code>verbose</code>	If TRUE, print verbose messages. Default is TRUE.
<code>perm</code>	The number of permutations. Default is 1000.
<code>C.matrix</code>	The contrast matrix. Default is NULL, and it will be generated from the module list.
<code>p.adjust.method</code>	Method for p-value adjustment. Default is "BH".
<code>...</code>	Additional parameters to be passed to <a href="#">safe</a> function.

**Value**

A data frame containing SAFE results for KO enrichment.

**See Also**

Other common\_enrich: [KO\\_enrich\(\)](#), [KO\\_fisher\(\)](#), [KO\\_gsa\(\)](#), [KO\\_gsea\(\)](#), [KO\\_gsva\(\)](#), [KO\\_padog\(\)](#), [KO\\_sea\(\)](#), [plot\\_enrich\\_res\(\)](#)

---

KO_sea	<i>Perform Simultaneous Enrichment Analysis</i>
--------	---

---

**Description**

Perform Simultaneous Enrichment Analysis

**Usage**

```
KO_sea(reporter_res, verbose = TRUE, ...)
```

**Arguments**

reporter_res	The input reporter result.
verbose	If TRUE, print verbose messages. Default is TRUE.
...	Additional parameters to be passed to <a href="#">SEA</a> function.

**Value**

enrich\_res

**See Also**

Other common\_enrich: [KO\\_enrich\(\)](#), [KO\\_fisher\(\)](#), [KO\\_gsa\(\)](#), [KO\\_gsea\(\)](#), [KO\\_gsva\(\)](#), [KO\\_padog\(\)](#), [KO\\_safe\(\)](#), [plot\\_enrich\\_res\(\)](#)

**Examples**

```
## use `SEA` from the `rSEA` package.  
if (requireNamespace("rSEA")) {  
  data("reporter_score_res")  
  sea_res <- KO_sea(reporter_score_res, verbose = TRUE)  
}
```

---

load_CARDinfo	<i>Load the CARDinfo (from CARD database)</i>
---------------	---

---

**Description**

Load the CARDinfo (from CARD database)

**Usage**

```
load_CARDinfo(verbose = TRUE)
```

**Arguments**

verbose            logical

**Value**

CARDinfo

---

load\_CAZy\_info            *Load the CAZy\_info (from CAZy database)*

---

**Description**

Load the CAZy\_info (from CAZy database)

**Usage**

```
load_CAZy_info(verbose = TRUE)
```

**Arguments**

verbose            logical

**Value**

CAZy\_info

---

load\_Enzyme\_info            *Load the Enzyme\_info (from ExPASy Enzyme database)*

---

**Description**

Load the Enzyme\_info (from ExPASy Enzyme database)

**Usage**

```
load_Enzyme_info(verbose = TRUE)
```

**Arguments**

verbose            logical

**Value**

Enzyme\_info

---

load_GOlist	<i>Load the GOlist (from 'GO' database)</i>
-------------	---

---

**Description**

Load the GOlist (from 'GO' database)

Load the GOinfo (from GO)

**Usage**

```
load_GOlist(verbose = TRUE)
```

```
load_GOinfo(verbose = TRUE)
```

**Arguments**

verbose            logical

**Value**

GOlist

GOinfo

---

load_htable	<i>Load the specific table (from 'KEGG')</i>
-------------	--

---

**Description**

Load the specific table (from 'KEGG')

Load the KOList (from 'KEGG')

Load the CPDlist (from 'KEGG')

Load the KO description (from 'KEGG')

Load the KO\_htable (from 'KEGG')

Load the Pathway\_htable (from 'KEGG')

Load the Enzyme\_htable (from 'KEGG')

Load the Module\_htable (from 'KEGG')

Load the Compound\_htable (from 'KEGG')

Load the pathway information for an organism (from 'KEGG')

**Usage**

```
load_htable(type, verbose = TRUE)

load_KOlist(verbose = TRUE)

load_CPDlist(verbose = TRUE)

load_KO_desc(verbose = TRUE)

load_KO_htable(verbose = TRUE)

load_Pathway_htable(verbose = TRUE)

load_Enzyme_htable(verbose = TRUE)

load_Module_htable(verbose = TRUE)

load_Compound_htable(verbose = TRUE)

load_org_pathway(org = "hsa", verbose = TRUE)
```

**Arguments**

type	"ko", "module", "pathway", "compound" ...
verbose	logical
org	kegg organism, listed in <a href="https://www.genome.jp/kegg/catalog/org_list.html">https://www.genome.jp/kegg/catalog/org_list.html</a> , default, "hsa"

**Value**

```
KO_htable
KOlist
CPDlist
KO description
KO_htable
Pathway_htable
Enzyme_htable
Module_htable
Compound_htable
KOlist
```

**Examples**

```
Pathway_htable <- load_htable("pathway")
head(Pathway_htable)
```

---

load\_pathway\_xml\_ls     *Load KEGG pathway XML file list*

---

### Description

Load KEGG pathway XML file list

### Usage

```
load_pathway_xml_ls(org = NULL, verbose = TRUE)
```

### Arguments

org                    kegg organism, listed in [https://www.genome.jp/kegg/catalog/org\\_list.html](https://www.genome.jp/kegg/catalog/org_list.html) such as "hsa", default NULL means ko.

verbose                Logical, whether to print messages about the loading process. Default is 'TRUE'.

### Value

A list of KEGG pathway XML files, where each element is a 'tbl\_graph' or 'igraph' object.

---

mmu\_kegg\_pathway     *pathway information for "mmu"*

---

### Description

pathway information for "mmu"

### See Also

Other data: [CPDlist](#), [Compound\\_htable](#), [Golist](#), [KO\\_htable](#), [Kolist](#), [Module\\_htable](#), [Pathway\\_htable](#), [hsa\\_kegg\\_pathway](#)

---

modify_description	<i>Modify the pathway description before plotting</i>
--------------------	---

---

**Description**

Modify the pathway description before plotting

**Usage**

```
modify_description(  
  reporter_res,  
  pattern = " - Homo sapiens (human)",  
  replacement = ""  
)
```

**Arguments**

reporter_res	reporter_res
pattern	str, like " - Homo sapiens (human)"
replacement	str, like ""

**Value**

reporter\_res

**Examples**

```
data("reporter_score_res")  
modify_description(reporter_score_res, pattern = " - Homo sapiens (human)")
```

---

Module_htable	<i>Module htable from 'KEGG'</i>
---------------	----------------------------------

---

**Description**

Module htable from 'KEGG'

**See Also**

Other data: [CPDlist](#), [Compound\\_htable](#), [G0list](#), [KO\\_htable](#), [K0list](#), [Pathway\\_htable](#), [hsa\\_kegg\\_pathway](#), [mmu\\_kegg\\_pathway](#)

---

parse_enzyme_dat	<i>Parse enzyme.dat file into a structured data frame</i>
------------------	---

---

**Description**

Parse enzyme.dat file into a structured data frame

**Usage**

```
parse_enzyme_dat(file_path)
```

**Arguments**

file\_path      Path to the enzyme.dat file.

**Value**

A data frame with columns: EC, Name, Aliases, Reaction, Comments, SwissProt\_Refs.

---

Pathway_hhtable	<i>Pathway htable from 'KEGG'</i>
-----------------	-----------------------------------

---

**Description**

Pathway htable from 'KEGG'

**See Also**

Other data: [CPDlist](#), [Compound\\_hhtable](#), [G0list](#), [KO\\_hhtable](#), [K0list](#), [Module\\_hhtable](#), [hsa\\_kegg\\_pathway](#), [mmu\\_kegg\\_pathway](#)

---

pathway_net_index	<i>Create a pathway network index data.frame</i>
-------------------	--

---

**Description**

Create a pathway network index data.frame

**Usage**

```
pathway_net_index(path_net_c)
```

**Arguments**

path\_net\_c      A 'tbl\_graph' or 'igraph' object representing a KEGG pathway network.

**Value**

A data frame containing vertex indices and their attributes, including in-degree and out-degree.

**See Also**

plot\_pathway\_net

---

plot.cm_res	<i>Plot c_means result</i>
-------------	----------------------------

---

**Description**

Plot c\_means result

**Usage**

```
## S3 method for class 'cm_res'
plot(
  x,
  filter_membership,
  mode = 1,
  show.clust.cent = TRUE,
  show_num = TRUE,
  ...
)
```

**Arguments**

x	a cm_res object
filter_membership	filter membership
mode	1~2
show.clust.cent	show cluster center?
show_num	show number of each cluster?
...	additional

**Value**

ggplot

---

plot_enrich_res	<i>Plot enrich_res</i>
-----------------	------------------------

---

### Description

Plot enrich\_res

Plot enrich\_res

### Usage

```
plot_enrich_res(  
  enrich_res,  
  mode = 1,  
  padj_threshold = 0.05,  
  show_ID = FALSE,  
  Pathway_description = TRUE,  
  facet_level = FALSE,  
  facet_anno = NULL,  
  str_width = 50,  
  facet_str_width = 15,  
  ...  
)  
  
## S3 method for class 'enrich_res'  
plot(  
  x,  
  mode = 1,  
  padj_threshold = 0.05,  
  show_ID = FALSE,  
  Pathway_description = TRUE,  
  facet_level = FALSE,  
  facet_anno = NULL,  
  str_width = 50,  
  facet_str_width = 15,  
  ...  
)
```

### Arguments

enrich_res	enrich_res object
mode	plot style: 1~2
padj_threshold	p.adjust threshold
show_ID	show pathway id
Pathway_description	show KO description rather than KO id.

facet_level	facet plot if the type is "pathway" or "module"
facet_anno	annotation table for facet, two columns, first is level summary, second is pathway id.
str_width	default: 50
facet_str_width	str width for facet label
...	add
x	enrich_res object

**Value**

ggplot  
ggplot

**See Also**

Other common\_enrich: [KO\\_enrich\(\)](#), [KO\\_fisher\(\)](#), [KO\\_gsa\(\)](#), [KO\\_gsea\(\)](#), [KO\\_gsva\(\)](#), [KO\\_padog\(\)](#), [KO\\_safe\(\)](#), [KO\\_sea\(\)](#)

---

plot\_features\_box      *Plot features boxplot*

---

**Description**

Plot features boxplot

**Usage**

```
plot_features_box(
  kof,
  group = NULL,
  metadata = NULL,
  map_id = "map00780",
  select_ko = NULL,
  only_sig = FALSE,
  box_param = NULL,
  modulelist = NULL,
  KO_description = FALSE,
  str_width = 50
)
```

**Arguments**

kodf	KO_abundance table, rowname is ko id (e.g. K00001), colnames is samples. or result of 'get_reporter_score'
group	The compare group (two category) in your data, one column name of metadata when metadata exist or a vector whose length equal to columns number of kodf.
metadata	metadata
map_id	the pathway or module id
select_ko	select which ko
only_sig	only show the significant features
box_param	parameters pass to <a href="#">group_box</a>
modulelist	NULL or customized modulelist dataframe, must contain "id", "K_num", "KOs", "Description" columns. Take the 'Kolist' as example, use <a href="#">custom_modulelist</a> .
KO_description	show KO description rather than KO id.
str_width	str_width to wrap

**Value**

ggplot

**Examples**

```

data("reporter_score_res")
plot_features_box(reporter_score_res,
  select_ko = c("K00059", "K00208", "K00647", "K00652", "K00833", "K01012"),
  box_param = list(p_value1 = FALSE, trend_line = TRUE)
)
plot_features_box(reporter_score_res,
  select_ko = "K00059", KO_description = TRUE,
  box_param = list(p_value1 = FALSE, trend_line = TRUE)
)

```

---

plot\_features\_distribution

*plot the Z-score of features distribution*

---

**Description**

plot the Z-score of features distribution

**Usage**

```
plot_features_distribution(
  reporter_res,
  map_id,
  text_size = 4,
  text_position = NULL,
  rug_length = 0.04
)
```

**Arguments**

reporter_res	result of 'reporter_score'
map_id	the pathway or module id
text_size	text_size=4
text_position	text_position, e.g. c(x=3,y=0.4)
rug_length	rug_length=0.04

**Value**

ggplot

**Examples**

```
data("reporter_score_res")
plot_features_distribution(reporter_score_res, map_id = c("map05230", "map03010"))
```

---

plot\_features\_heatmap *Plot features heatmap*

---

**Description**

Plot features heatmap

**Usage**

```
plot_features_heatmap(
  kofc,
  group = NULL,
  metadata = NULL,
  map_id = "map00780",
  select_ko = NULL,
  only_sig = FALSE,
  columns = NULL,
  modulelist = NULL,
  KO_description = FALSE,
```

```

    str_width = 50,
    heatmap_param = list()
  )

```

### Arguments

kodf	KO_abundance table, rowname is ko id (e.g. K00001), colnames is samples. or result of 'get_reporter_score'
group	The compare group (two category) in your data, one column name of metadata when metadata exist or a vector whose length equal to columns number of kodf.
metadata	metadata
map_id	the pathway or module id
select_ko	select which ko
only_sig	only show the significant KOs
columns	change columns
modulelist	NULL or customized modulelist dataframe, must contain "id", "K_num", "KOs", "Description" columns. Take the 'Kolist' as example, use <a href="#">custom_modulelist</a> .
KO_description	show KO description rather than KO id.
str_width	str_width to wrap
heatmap_param	parameters pass to <a href="#">pheatmap</a>

### Value

ggplot

### Examples

```

if (requireNamespace("pheatmap")) {
  data("reporter_score_res")
  plot_features_heatmap(reporter_score_res, map_id = "map00780")
}

```

---

plot\_features\_in\_pathway

*Plot features trend in one pathway or module*

---

### Description

Plot features trend in one pathway or module

**Usage**

```
plot_features_in_pathway(
  ko_stat,
  map_id = "map00780",
  modulelist = NULL,
  select_ko = NULL,
  box_color = reporter_color,
  show_number = TRUE,
  scale = FALSE,
  feature_type = "KOs",
  line_color = c(Depleted = "seagreen", Enriched = "orange", None = "grey", Significant =
    "red2")
)
```

**Arguments**

ko_stat	ko_stat result from <a href="#">pvalue2zs</a> or result of 'get_reporter_score'
map_id	the pathway or module id
modulelist	NULL or customized modulelist dataframe, must contain "id","K_num","KOs","Description" columns. Take the 'KOList' as example, use <a href="#">custom_modulelist</a> .
select_ko	select which ko
box_color	box and point color, default: c("#e31a1c","#1f78b4")
show_number	show the numbers.
scale	scale the data by row.
feature_type	show in the title ,default: KOs
line_color	line color, default: c("Depleted"="seagreen","Enriched"="orange","None"="grey")

**Value**

ggplot

**Examples**

```
data("reporter_score_res")
plot_features_in_pathway(ko_stat = reporter_score_res, map_id = "map00860")
```

---

plot\_features\_network *Plot features network*

---

**Description**

Plot features network

**Usage**

```

plot_features_network(
  ko_stat,
  map_id = "map00780",
  near_pathway = FALSE,
  modulelist = NULL,
  kos_color = c(Depleted = "seagreen", Enriched = "orange", None = "grey", Significant =
    "red2", Pathway = "#80b1d3"),
  pathway_label = TRUE,
  kos_label = TRUE,
  pathway_description = FALSE,
  kos_description = FALSE,
  str_width = 50,
  mark_module = FALSE,
  mark_color = NULL,
  return_net = FALSE,
  ...
)

```

**Arguments**

ko_stat	ko_stat result from <a href="#">pvalue2zs</a> or result of 'get_reporter_score'
map_id	the pathway or module id
near_pathway	show the near_pathway if any features exist.
modulelist	NULL or customized modulelist dataframe, must contain "id", "K_num", "KOs", "Description" columns. Take the 'KOlister' as example, use <a href="#">custom_modulelist</a> .
kos_color	default, c("Depleted"="seagreen", "Enriched"="orange", "None"="grey", "Significant"="red2")
pathway_label	show pathway_label?
kos_label	show kos_label?
pathway_description	show the pathway description?
kos_description	show the kos description?
str_width	str width
mark_module	mark the modules?
mark_color	mark colors, default, c("Depleted"="seagreen", "Enriched"="orange", "None"="grey", "Significant"="red2")
return_net	return the network
...	additional arguments for <a href="#">c_net_plot</a>

**Value**

network plot

**Examples**

```

if (requireNamespace("MetaNet")) {
  data("reporter_score_res")
  plot_features_network(reporter_score_res, map_id = "map05230")
  plot_features_network(reporter_score_res, map_id = "map00780", near_pathway = TRUE)
}

```

---

plot\_htable

*Plot htable levels*


---

**Description**

Plot htable levels

**Usage**

```
plot_htable(type = "ko", select = NULL, htable = NULL)
```

**Arguments**

type	"ko", "module", "pathway", "compound"
select	select ids
htable	custom a htable

**Value**

ggplot

**Examples**

```

data("KO_abundance_test")
plot_htable(select = rownames(KO_abundance))

```

---

plot\_KEGG\_map

*plot\_KEGG\_map*


---

**Description**

plot\_KEGG\_map

**Usage**

```
plot_KEGG_map(
  ko_stat,
  map_id = "map00780",
  modulelist = NULL,
  type = "pathway",
  feature = "ko",
  color_var = "Z_score",
  save_dir,
  color = c("seagreen", "grey", "orange")
)
```

**Arguments**

ko_stat	ko_stat result from <a href="#">pvalue2zs</a> or result of 'get_reporter_score'
map_id	the pathway or module id
modulelist	NULL or customized modulelist dataframe, must contain "id", "K_num", "KOs", "Description" columns. Take the 'KOl原因' as example, use <a href="#">custom_modulelist</a> .
type	"pathway" or "module" for default KOl原因 for microbiome, "CC", "MF", "BP", "ALL" for default GOl原因 for homo sapiens. And org in listed in 'https://www.genome.jp/kegg/catalog/org' such as "hsa" (if your kodf is come from a specific organism, you should specify type here).
feature	one of "ko", "gene", "compound"
color_var	use which variable to color
save_dir	where to save the png files
color	color

**Value**

png files

**References**

<https://zhuanlan.zhihu.com/p/357687076>

**Examples**

```
message("The following example will download some files, run yourself:")

if (requireNamespace("pathview")) {
  output_dir <- tempdir()
  data("reporter_score_res")
  plot_KEGG_map(reporter_score_res$ko_stat,
    map_id = "map00780", type = "pathway",
    feature = "ko", color_var = "Z_score", save_dir = output_dir
  )
}
```

---

plot\_pathway\_net      *Plot a KEGG pathway network*

---

### Description

Plot a KEGG pathway network

### Usage

```
plot_pathway_net(path_net_c, simplify = FALSE, plot_depth = FALSE, ...)
```

### Arguments

path_net_c	A 'metanet' object representing the pathway network.
simplify	Logical, whether to simplify the network by removing multiple edges and loops. Default is 'FALSE'.
plot_depth	Logical, whether to plot the network as a tree layout. Default is 'FALSE'.
...	Additional arguments passed to 'MetaNet::c_net_plot'.

### Value

A plot of the pathway network.

### Examples

```
if (requireNamespace("MetaNet") && requireNamespace("ggkegg")) {
  tmp_dir <- tempdir()
  pcutils::download2("https://rest.kegg.jp/get/ko01521/kgml", file.path(tmp_dir, "ko01521.xml"))
  path_net_c <- c_net_from_pathway_xml(file.path(tmp_dir, "ko01521.xml"))
  plot_pathway_net(path_net_c)
  pathway_net_index(path_net_c)
}
```

---

plot\_report      *Plot the reporter\_res*

---

### Description

Plot the reporter\_res

**Usage**

```
plot_report(
  reporter_res,
  rs_threshold = 1.64,
  mode = 1,
  y_text_size = 13,
  str_width = 100,
  show_ID = FALSE,
  Pathway_description = TRUE,
  facet_level = FALSE,
  facet_anno = NULL,
  facet_str_width = 15,
  plot_line = TRUE,
  reorder = FALSE
)
```

**Arguments**

reporter_res	result of 'get_reporter_score' or 'reporter_score'
rs_threshold	plot threshold vector, default:1.64
mode	1~3 plot style.
y_text_size	y_text_size
str_width	str_width to wrap
show_ID	show pathway id
Pathway_description	show KO description rather than KO id.
facet_level	facet plot if the type is "pathway" or "module"
facet_anno	annotation table for facet, two columns, first is level summary, second is pathway id.
facet_str_width	str width for facet label
plot_line	plot line or not
reorder	reorder the order of the pathways

**Value**

ggplot

**Examples**

```
data("reporter_score_res")
plot_report(reporter_score_res, rs_threshold = c(2.5, -2.5), y_text_size = 10, str_width = 40)
```

---

`plot_report_circle_packing`*Plot the reporter\_res as circle\_packing*

---

### Description

Plot the reporter\_res as circle\_packing

### Usage

```
plot_report_circle_packing(  
  reporter_res,  
  rs_threshold = 1.64,  
  mode = 2,  
  facet_anno = NULL,  
  show_ID = FALSE,  
  Pathway_description = TRUE,  
  str_width = 10,  
  show_level_name = "all",  
  show_tip_label = TRUE  
)
```

### Arguments

reporter_res	result of 'get_reporter_score'
rs_threshold	plot threshold vector, default:1.64
mode	1~2 plot style.
facet_anno	annotation table for facet, more two columns, last is pathway name, last second is pathway id.
show_ID	show pathway id
Pathway_description	show KO description rather than KO id.
str_width	str_width to wrap
show_level_name	show the level name?
show_tip_label	show the tip label?

### Value

ggplot

**Examples**

```
data("reporter_score_res")
if (requireNamespace("igraph") && requireNamespace("ggraph")) {
  plot_report_circle_packing(reporter_score_res, rs_threshold = c(2, -2), str_width = 40)
}
```

---

plot\_significance      *Plot the significance of pathway*

---

**Description**

Plot the significance of pathway

**Usage**

```
plot_significance(reporter_res, map_id)
```

**Arguments**

reporter\_res      result of 'get\_reporter\_score' or 'reporter\_score'  
map\_id            the pathway or module id

**Value**

ggplot

**Examples**

```
data("reporter_score_res")
plot_significance(reporter_score_res, map_id = c("map05230", "map03010"))
```

---

print.reporter\_score      *Print reporter\_score*

---

**Description**

Print reporter\_score

**Usage**

```
## S3 method for class 'reporter_score'
print(x, ...)
```

**Arguments**

x	reporter_score
...	add

**Value**

No value

---

print.rs_by_cm	<i>Print rs_by_cm</i>
----------------	-----------------------

---

**Description**

Print rs\_by\_cm

**Usage**

```
## S3 method for class 'rs_by_cm'
print(x, ...)
```

**Arguments**

x	rs_by_cm
...	add

**Value**

No value

---

pvalue2zs	<i>Transfer p-value of KOs to Z-score</i>
-----------	---

---

**Description**

Transfer p-value of KOs to Z-score

**Usage**

```
pvalue2zs(
  ko_pvalue,
  mode = c("directed", "mixed")[1],
  p.adjust.method1 = "none"
)
```

## Arguments

ko_pvalue	data.frame from <a href="#">ko.test</a> , 'KO_id' and 'p.value' columns are required.
mode	'mixed' or 'directed' (default, only for two groups differential analysis or multi-groups correlation analysis.), see details in <a href="#">pvalue2zs</a> .
p.adjust.method1	p.adjust.method for 'ko.test', see <a href="#">p.adjust</a>

## Details

'mixed' mode is the original reporter-score method from Patil, K. R. et al. PNAS 2005. In this mode, the reporter score is **undirected**, and the larger the reporter score, the more significant the enrichment, but it cannot indicate the up-and-down regulation information of the pathway! (Liu, L. et al. iMeta 2023.)

steps:

1. Use the Wilcoxon rank sum test to obtain the P value of the significance of each KO difference between the two groups (ie  $P_{koi}$ , i represents a certain KO);
2. Using an inverse normal distribution, convert the P value of each KO into a Z value ( $Z_{koi}$ ), the formula:

$$Z_{koi} = \theta^{-1}(1 - P_{koi})$$

3. 'Upgrade' KO to pathway:  $Z_{koi}$ , calculate the Z value of the pathway, the formula:

$$Z_{pathway} = \frac{1}{\sqrt{k}} \sum Z_{koi}$$

where k means A total of k KOs were annotated to the corresponding pathway;

4. Evaluate the degree of significance: permutation (permutation) 1000 times, get the random distribution of  $Z_{pathway}$ , the formula:

$$Z_{adjustedpathway} = (Z_{pathway} - \mu_k) / \sigma_k$$

$\mu_k$  is The mean of the random distribution,  $\sigma_k$  is the standard deviation of the random distribution.

Instead, '**directed**' mode is a derived version of 'mixed', referenced from <https://github.com/wangpeng407/ReporterScore>

This approach is based on the same assumption of many differential analysis methods: the expression of most genes has no significant change.

steps:

1. Use the Wilcoxon rank sum test to obtain the P value of the significance of each KO difference between the two groups (ie  $P_{koi}$ , i represents a certain KO), and then divide the P value by 2, that is, the range of (0,1] becomes (0,0.5],  $P_{koi} = P_{koi}/2$ ;
2. Using an inverse normal distribution, convert the P value of each KO into a Z value ( $Z_{koi}$ ), the formula:

$$Z_{koi} = \theta^{-1}(1 - P_{koi})$$

since the above P value is less than 0.5, all Z values will be greater than 0;

3. Considering whether each KO is up-regulated or down-regulated, calculate  $diff\_KO$ ,

$$Z_{koi} = -Z_{koi} \quad (diff\_KO < 0),$$

so  $Z_{koi}$  is greater than 0 Up-regulation,  $Z_{koi}$  less than 0 is down-regulation;

4. 'Upgrade' KO to pathway:  $Z_{koi}$ , calculate the Z value of the pathway, the formula:

$$Z_{pathway} = \frac{1}{\sqrt{k}} \sum Z_{koi}$$

where k means A total of k KOs were annotated to the corresponding pathway;

5. Evaluate the degree of significance: permutation (permutation) 1000 times, get the random distribution of  $Z_{pathway}$ , the formula:

$$Z_{adjustedpathway} = (Z_{pathway} - \mu_k) / \sigma_k$$

$\mu_k$  is The mean of the random distribution,  $\sigma_k$  is the standard deviation of the random distribution.

The finally obtained  $Z_{adjustedpathway}$  is the Reporter score value enriched for each pathway. In this mode, the Reporter score is directed, and a larger positive value represents a significant up-regulation enrichment, and a smaller negative values represent significant down-regulation enrichment.

However, the disadvantage of this mode is that when a pathway contains about the same number of significantly up-regulates KOs and significantly down-regulates KOs, the final absolute value of Reporter score may approach 0, becoming a pathway that has not been significantly enriched.

## Value

ko\_stat data.frame

## References

1. Patil, K. R. & Nielsen, J. Uncovering transcriptional regulation of metabolism by using metabolic network topology. *Proc Natl Acad Sci U S A* 102, 2685–2689 (2005).
2. Liu, L., Zhu, R. & Wu, D. Misuse of reporter score in microbial enrichment analysis. *iMeta* n/a, e95.
3. <https://github.com/wangpeng407/Reporter>

## See Also

Other GRSA: [combine\\_rs\\_res\(\)](#), [get\\_reporter\\_score\(\)](#), [ko.test\(\)](#), [reporter\\_score\(\)](#)

## Examples

```
data("KO_abundance_test")
ko_pvalue <- ko.test(KO_abundance, "Group", metadata)
ko_stat <- pvalue2zs(ko_pvalue, mode = "directed")
```

---

reporter\_score

*One step to get the reporter score of your KO abundance table.*

---

## Description

One step to get the reporter score of your KO abundance table.

**Usage**

```
reporter_score(
  kodf,
  group,
  metadata = NULL,
  method = "wilcox.test",
  pattern = NULL,
  p.adjust.method1 = "none",
  mode = c("directed", "mixed")[1],
  verbose = TRUE,
  feature = "ko",
  type = c("pathway", "module")[1],
  p.adjust.method2 = "BH",
  modulelist = NULL,
  threads = 1,
  perm = 4999,
  min_exist_KO = 3,
  max_exist_KO = 600
)
```

**Arguments**

kodf	KO_abundance table, rowname are feature ids (e.g. K00001 if feature="ko"; PEX11A if feature="gene"; C00024 if feature="compound"), colnames are samples.
group	The comparison groups (at least two categories) in your data, one column name of metadata when metadata exist or a vector whose length equal to columns number of kodf. And you can use factor levels to change order.
metadata	sample information data.frame contains group
method	the type of test. Default is 'wilcox.test'. Allowed values include: <ul style="list-style-type: none"> <li>• <a href="#">t.test</a> (parametric) and <a href="#">wilcox.test</a> (non-parametric). Perform comparison between two groups of samples. If the grouping variable contains more than two levels, then a pairwise comparison is performed.</li> <li>• <a href="#">anova</a> (parametric) and <a href="#">kruskal.test</a> (non-parametric). Perform one-way ANOVA test comparing multiple groups.</li> <li>• 'pearson', 'kendall', or 'spearman' (correlation), see <a href="#">cor</a>.</li> </ul>
pattern	a named vector matching the group, e.g. c('G1'=1,'G2'=3,'G3'=2), use the correlation analysis with specific pattern to calculate p-value.
p.adjust.method1	p.adjust.method for 'ko.test', see <a href="#">p.adjust</a>
mode	'mixed' or 'directed' (default, only for two groups differential analysis or multi-groups correlation analysis.), see details in <a href="#">pvalue2zs</a> .
verbose	logical
feature	one of 'ko', 'gene', 'compound'

type	'pathway' or 'module' for default Kolist for microbiome, 'CC', 'MF', 'BP', 'ALL' for default Golist for homo sapiens. And org in listed in 'https://www.genome.jp/kegg/catalog/org_' such as 'hsa' (if your kodf is come from a specific organism, you should specify type here).
p.adjust.method2	p.adjust.method for the correction of ReporterScore, see <a href="#">p.adjust</a>
modulelist	NULL or customized modulelist dataframe, must contain 'id','K_num','KOs','Description' columns. Take the 'Kolist' as example, use <a href="#">custom_modulelist</a> .
threads	default 1
perm	permutation number, default: 4999.
min_exist_KO	min exist KO number in a pathway (default, 3, when a pathway contains KOs less than 3, there will be no RS)
max_exist_KO	max exist KO number in a pathway (default, 600, when a pathway contains KOs more than 600, there will be no RS)

### Value

reporter\_score object:

kodf	your input KO_abundance table
ko_stat	ko statistics result contains p.value and z_score
reporter_s	the reporter score in each pathway
modulelist	default Kolist or customized modulelist dataframe
group	The comparison groups in your data
metadata	sample information dataframe contains group

for the 'reporter\_s' in result, whose columns represent:

ID	pathway id
Description	pathway description
K_num	total number of KOs/genes in the pathway
Exist_K_num	number of KOs/genes in your inputdata that exist in the pathway
Significant_K_num	number of kos/genes in your inputdata that are significant in the pathway
Z_score	$Z_{pathway} = \frac{1}{\sqrt{k}} \sum Z_{koi}$
BG_Mean	Background mean, $\mu_k$
BG_Sd	Background standard deviation, $\sigma_k$
ReporterScore	ReporterScore of the pathway, $ReporterScore = (Z_{pathway} - \mu_k) / \sigma_k$
p.value	p.value of the ReporterScore
p.adjust	adjusted p.value by p.adjust.method2

### See Also

Other GRSA: [combine\\_rs\\_res\(\)](#), [get\\_reporter\\_score\(\)](#), [ko.test\(\)](#), [pvalue2zs\(\)](#)

## Examples

```
message("The following example require some time to run:")

data("KO_abundance_test")
reporter_score_res <- reporter_score(KO_abundance, "Group", metadata,
  mode = "directed", perm = 499
)
head(reporter_score_res$reporter_s)
reporter_score_res2 <- reporter_score(KO_abundance, "Group2", metadata,
  mode = "mixed",
  method = "kruskal.test", p.adjust.method1 = "none", perm = 499
)
reporter_score_res3 <- reporter_score(KO_abundance, "Group2", metadata,
  mode = "directed",
  method = "pearson", pattern = c("G1" = 1, "G2" = 3, "G3" = 2), perm = 499
)
```

---

reporter\_score\_res      *'reporter\_score()' result from KO\_abundance\_test*

---

## Description

'reporter\_score()' result from KO\_abundance\_test

'reporter\_score()' result from KO\_abundance\_test

## Format

a list contain 7 elements.

**kodf** your input KO\_abundance table

**ko\_stat** ko statistics result contains p.value and z\_score

**reporter\_s** the reporter score in each pathway

**modulelist** default KOlist or customized modulelist dataframe

**group** The compare group (two category) in your data

**metadata** sample information dataframe contains group

## See Also

Other test\_data: [KO\\_abundance](#), [genedf](#)

RSA\_by\_cm

*Reporter score analysis after C-means clustering***Description**

Reporter score analysis after C-means clustering

Extract one cluster from rs\_by\_cm object

Plot c\_means result

**Usage**

```
RSA_by_cm(
  kodf,
  group,
  metadata = NULL,
  k_num = NULL,
  filter_var = 0.7,
  verbose = TRUE,
  method = "pearson",
  ...
)

extract_cluster(rsa_cm_res, cluster = 1)

plot_c_means(
  rsa_cm_res,
  filter_membership,
  mode = 1,
  show.clust.cent = TRUE,
  show_num = TRUE,
  ...
)
```

**Arguments**

kodf	KO_abundance table, rowname is ko id (e.g. K00001), colnames is samples.
group	The comparison groups (at least two categories) in your data, one column name of metadata when metadata exist or a vector whose length equal to columns number of kodf. And you can use factor levels to change order.
metadata	sample information data.frame contains group
k_num	if NULL, perform the cm_test_k, else an integer
filter_var	see c_means
verbose	verbose
method	method from <a href="#">reporter_score</a>

```

...          additional
rsa_cm_res   a cm_res object
cluster      integer
filter_membership
              filter membership 0~1.
mode         1~2
show.clust.cent
              show cluster center?
show_num     show number of each cluster?

```

**Value**

```

rs_by_cm
reporter_score object
ggplot

```

**See Also**

Other C\_means: [cm\\_test\\_k\(\)](#)

**Examples**

```

message("The following example require some time to run:")

if (requireNamespace("e1071") && requireNamespace("factoextra")) {
  data("KO_abundance_test")
  rsa_cm_res <- RSA_by_cm(KO_abundance, "Group2", metadata,
    k_num = 3,
    filter_var = 0.7, method = "pearson", perm = 199
  )
  extract_cluster(rsa_cm_res, cluster = 1)
}

```

---

```

update_CARDinfo      update CARDinfo from (from 'CARD' database)

```

---

**Description**

update CARDinfo from (from 'CARD' database)

**Usage**

```
update_CARDinfo(download_dir = NULL, card_data = NULL)
```

**Arguments**

download\_dir    download\_dir  
 card\_data        card\_data from <https://card.mcmaster.ca/download/0/broadstreet-v4.0.0.tar.bz2>

**Value**

No value

---

update_CAZy_info	<i>Update CAZy_info from CAZy database</i>
------------------	--

---

**Description**

Update CAZy\_info from CAZy database

**Usage**

```
update_CAZy_info(  
  download_dir = NULL,  
  fam_activities_file = NULL,  
  fam_subfam_ec_file = NULL  
)
```

**Arguments**

download\_dir    Directory to download the CAZy data files. If NULL, a temporary directory will be used.  
 fam\_activities\_file    the CAZy family activities file.  
 fam\_subfam\_ec\_file    the CAZy family-subfamily-EC file.

**Value**

No value

---

update_Enzyme_info	<i>Update Enzyme_info from ExPASy Enzyme database</i>
--------------------	---

---

**Description**

Update Enzyme\_info from ExPASy Enzyme database

**Usage**

```
update_Enzyme_info(  
    download_dir = NULL,  
    enzyme_dat_file = NULL,  
    enzclass_file = NULL  
)
```

**Arguments**

download_dir	Directory to download the Enzyme data files. If NULL, a temporary directory will be used.
enzyme_dat_file	Path to the manually downloaded 'enzyme.dat' file. If NULL, the file will be downloaded.
enzclass_file	Path to the manually downloaded 'enzclass.txt' file. If NULL, the file will be downloaded.

**Value**

No value

---

update_GOlist	<i>Update the GO2gene files (from 'GO' database)</i>
---------------	--

---

**Description**

Download links: <http://geneontology.org/docs/download-ontology/> <https://asa12138.github.io/FileList/GO>

**Usage**

```
update_GOlist(download_dir = NULL, GO_file = NULL)  
  
update_GOinfo(download_dir = NULL, obo_file = NULL)
```

**Arguments**

download_dir	download_dir
GO_file	GO_file
obo_file	obo_file from <a href="http://current.geneontology.org/ontology/go.obo">http://current.geneontology.org/ontology/go.obo</a>

**Value**

No value

---

update_KEGG	<i>Update files from 'KEGG'</i>
-------------	---------------------------------

---

**Description**

Download links:

<https://rest.kegg.jp/list/pathway> <https://rest.kegg.jp/link/pathway/ko> <https://rest.kegg.jp/link/pathway/hsa>  
<https://rest.kegg.jp/list/module> <https://rest.kegg.jp/link/module/ko> <https://rest.kegg.jp/link/module/hsa>

**Usage**

```
update_KEGG(download_dir)

update_KO_file(download_dir, RDSfile = NULL)

update_htable(type, keg_file = NULL, download = FALSE, download_dir = NULL)

update_org_pathway(
  org = "hsa",
  RDS_file = NULL,
  download = TRUE,
  download_dir = NULL
)
```

**Arguments**

download_dir	where to save the .keg file?
RDSfile	saved KO_files.RDS file
type	"ko", "module", "pathway", "compound" ...
keg_file	path of a .keg file, such as ko00001.keg from <a href="https://www.genome.jp/kegg-bin/download_htext?htext=ko00001&amp;format=htext">https://www.genome.jp/kegg-bin/download_htext?htext=ko00001&amp;format=htext</a> .
download	save the .keg file?
org	kegg organism, listed in <a href="https://www.genome.jp/kegg/catalog/org_list.html">https://www.genome.jp/kegg/catalog/org_list.html</a> , default, "hsa"
RDS_file	path of a org.RDS file if you saved before.

**Value**

No value

---

update\_pathway\_xml\_ls *Download KEGG pathway XML files and create networks*

---

**Description**

Download KEGG pathway XML files and create networks

**Usage**

```
update_pathway_xml_ls(download_dir, org = NULL)
```

**Arguments**

download_dir	Directory to save the downloaded XML files.
org	kegg organism, listed in <a href="https://www.genome.jp/kegg/catalog/org_list.html">https://www.genome.jp/kegg/catalog/org_list.html</a> such as "hsa", default NULL means ko.

**Value**

No value

---

up\_level\_KO *Upgrade the KO level*

---

**Description**

Upgrade the KO level

**Usage**

```
up_level_KO(  
  KO_abundance,  
  level = "pathway",  
  show_name = FALSE,  
  modulelist = NULL,  
  verbose = TRUE  
)
```

**Arguments**

KO_abundance	KO_abundance
level	one of 'pathway', 'module', 'level1', 'level2', 'level3', 'module1', 'module2', 'module3'.
show_name	logical
modulelist	NULL or customized modulelist dataframe, must contain 'id', 'K_num', 'KOs', 'Description' columns. Take the 'KOlist' as example, use <a href="#">custom_modulelist</a> .
verbose	logical

**Value**

data.frame

**Examples**

```
data("KO_abundance_test")
KO_level1 <- up_level_KO(KO_abundance, level = "level1", show_name = TRUE)
```

# Index

- \* **C\_means**
  - cm\_test\_k, 3
  - RSA\_by\_cm, 50
- \* **GRSA**
  - combine\_rs\_res, 4
  - get\_reporter\_score, 11
  - ko.test, 13
  - pvalue2zs, 44
  - reporter\_score, 46
- \* **common\_enrich**
  - KO\_enrich, 15
  - KO\_fisher, 16
  - KO\_gsa, 17
  - KO\_gsea, 18
  - KO\_gsva, 20
  - KO\_padog, 21
  - KO\_safe, 22
  - KO\_sea, 23
  - plot\_enrich\_res, 31
- \* **data**
  - Compound\_htable, 5
  - CPDlist, 5
  - GOList, 12
  - hsa\_kegg\_pathway, 13
  - KO\_htable, 20
  - KOList, 14
  - mmu\_kegg\_pathway, 27
  - Module\_htable, 28
  - Pathway\_htable, 29
- \* **modulelist**
  - custom\_modulelist, 6
  - custom\_modulelist\_from\_org, 7
  - get\_features, 10
- \* **test\_data**
  - genedf, 9
  - KO\_abundance, 15
  - reporter\_score\_res, 49
- anova, 14, 47
- as.enrich\_res (KO\_enrich), 15
- c\_means (cm\_test\_k), 3
- c\_net\_from\_pathway\_xml, 8
- c\_net\_plot, 37
- cm\_test\_k, 3, 51
- combine\_rs\_res, 4, 12, 14, 46, 48
- Compound\_htable, 5, 5, 12–14, 20, 27–29
- cor, 14, 47
- CPDlist, 5, 5, 12–14, 20, 27–29
- custom\_modulelist, 4, 6, 7, 10, 11, 16, 17, 19, 33, 35–37, 39, 48, 56
- custom\_modulelist\_from\_org, 6, 7, 11
- download\_org\_pathway (update\_KEGG), 54
- export\_report\_table, 8
- extract\_cluster (RSA\_by\_cm), 50
- gene2ko, 9
- genedf, 9, 15, 49
- get\_all\_pathway\_net\_index, 10
- get\_features, 6, 7, 10
- get\_KOs (get\_features), 10
- get\_org\_pathway (update\_KEGG), 54
- get\_reporter\_score, 4, 11, 14, 46, 48
- GOList, 5, 12, 13, 14, 20, 27–29
- group\_box, 33
- GRSA (reporter\_score), 46
- GRSA\_by\_cm (RSA\_by\_cm), 50
- GSA, 18
- gsva, 20
- hsa\_kegg\_pathway, 5, 12, 13, 14, 20, 27–29
- ko.test, 4, 12, 13, 15, 17, 19, 20, 45, 46, 48
- KO\_abundance, 9, 15, 49
- KO\_enrich, 15, 17–23, 32
- KO\_fisher, 16, 16, 18–23, 32
- KO\_gsa, 16, 17, 17, 19–23, 32
- KO\_gsea, 16–18, 18, 20–23, 32
- KO\_gsva, 16–19, 20, 21–23, 32
- KO\_htable, 5, 12–14, 20, 27–29

- KO\_padog, [16–20](#), [21](#), [22](#), [23](#), [32](#)
- KO\_safe, [16–21](#), [22](#), [23](#), [32](#)
- KO\_sea, [16–22](#), [23](#), [32](#)
- KOlist, [5](#), [12](#), [13](#), [14](#), [20](#), [27–29](#)
- kruskal.test, [14](#), [47](#)
  
- load\_CARDinfo, [23](#)
- load\_CAZy\_info, [24](#)
- load\_Compound\_htable (load\_htable), [25](#)
- load\_CPDlist (load\_htable), [25](#)
- load\_Enzyme\_htable (load\_htable), [25](#)
- load\_Enzyme\_info, [24](#)
- load\_GOinfo (load\_GOlist), [25](#)
- load\_GOlist, [25](#)
- load\_htable, [25](#)
- load\_KO\_desc (load\_htable), [25](#)
- load\_KO\_htable (load\_htable), [25](#)
- load\_KOlist (load\_htable), [25](#)
- load\_Module\_htable (load\_htable), [25](#)
- load\_org\_pathway (load\_htable), [25](#)
- load\_Pathway\_htable (load\_htable), [25](#)
- load\_pathway\_xml\_ls, [27](#)
  
- metadata (KO\_abundance), [15](#)
- mmu\_kegg\_pathway, [5](#), [12–14](#), [20](#), [27](#), [28](#), [29](#)
- modify\_description, [28](#)
- Module\_htable, [5](#), [12–14](#), [20](#), [27](#), [28](#), [29](#)
  
- p.adjust, [11](#), [14](#), [45](#), [47](#), [48](#)
- padog, [21](#)
- parse\_enzyme\_dat, [29](#)
- Pathway\_htable, [5](#), [12–14](#), [20](#), [27](#), [28](#), [29](#)
- pathway\_net\_index, [29](#)
- pheatmap, [35](#)
- plot.cm\_res, [30](#)
- plot.enrich\_res (plot\_enrich\_res), [31](#)
- plot\_c\_means (RSA\_by\_cm), [50](#)
- plot\_enrich\_res, [16–23](#), [31](#)
- plot\_features\_box, [32](#)
- plot\_features\_distribution, [33](#)
- plot\_features\_heatmap, [34](#)
- plot\_features\_in\_pathway, [35](#)
- plot\_features\_network, [36](#)
- plot\_htable, [38](#)
- plot\_KEGG\_map, [38](#)
- plot\_KOs\_box (plot\_features\_box), [32](#)
- plot\_KOs\_distribution
  - (plot\_features\_distribution), [33](#)
  
- plot\_KOs\_heatmap
  - (plot\_features\_heatmap), [34](#)
- plot\_KOs\_in\_pathway
  - (plot\_features\_in\_pathway), [35](#)
- plot\_KOs\_network
  - (plot\_features\_network), [36](#)
- plot\_pathway\_net, [40](#)
- plot\_report, [40](#)
- plot\_report\_bar (plot\_report), [40](#)
- plot\_report\_circle\_packing, [42](#)
- plot\_significance, [43](#)
- print.reporter\_score, [43](#)
- print.rs\_by\_cm, [44](#)
- pvalue2zs, [4](#), [10–12](#), [14](#), [36](#), [37](#), [39](#), [44](#), [45](#), [47](#), [48](#)
  
- reporter\_score, [4](#), [12](#), [14](#), [46](#), [46](#), [50](#)
- reporter\_score\_res, [9](#), [15](#), [49](#)
- reporter\_score\_res2
  - (reporter\_score\_res), [49](#)
- RSA (reporter\_score), [46](#)
- RSA\_by\_cm, [4](#), [50](#)
  
- safe, [22](#)
- SEA, [23](#)
  
- t.test, [13](#), [47](#)
- transform\_modulelist
  - (custom\_modulelist), [6](#)
  
- up\_level\_KO, [55](#)
- update\_CARDinfo, [51](#)
- update\_CAZy\_info, [52](#)
- update\_Enzyme\_info, [53](#)
- update\_GOinfo (update\_GOlist), [53](#)
- update\_GOlist, [53](#)
- update\_htable (update\_KEGG), [54](#)
- update\_KEGG, [54](#)
- update\_KO\_file (update\_KEGG), [54](#)
- update\_org\_pathway (update\_KEGG), [54](#)
- update\_pathway\_xml\_ls, [55](#)
  
- wilcox.test, [13](#), [47](#)