Package 'MetaNet'

May 26, 2026

Type Package

Title Network Analysis for Omics Data

Version 0.3.2

Description Comprehensive network analysis package.
Calculate correlation network fastly, accelerate lots of analysis by parallel computing.
Support for multi-omics data, search sub-nets fluently.
Handle bigger data, more than 10,000 nodes in each omics.
Offer various layout method for multi-omics network and some interfaces to other software ('Gephi', 'Cytoscape', 'ggplot2'), easy to visualize.
Provide comprehensive topology indexes calculation, including ecological network stability.

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 4.1.0), igraph (>= 1.3.5)

LazyData true

Imports graphics, dplyr, ggplot2 (>= 3.2.0), ggnewscale, ggrepel, grDevices, magrittr, reshape2, stats, tibble, utils, pcutils (>= 0.2.7), rlang

Suggests pheatmap, vegan, stringr, foreach, doSNOW, snow, knitr, rmarkdown, prettydoc, Hmisc, gifski, ggraph, networkD3, ggpmisc, ggtree, treeio, circlize, jsonify, ggpubr, corrplot, philentropy, spatstat.random, spatstat.geom, sf

VignetteBuilder knitr

BugReports https://github.com/Asa12138/MetaNet/issues

URL https://github.com/Asa12138/MetaNet

ByteCompile true

biocViews DataImport, Network analysis, Omics, Software, Visualization

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 2026-05-26 03:10:03 UTC

Contents

`adjacency_similarity` *Calculate Similarity Between Two Graphs via Adjacency Matrices*

Description

Computes the similarity between two igraph objects using their adjacency matrices. Supports Frobenius norm-based similarity and cosine similarity.

Usage

```
adjacency_similarity(g1, g2, method = "frobenius")
```

Arguments

`g1`	An igraph object representing the first graph.
`g2`	An igraph object representing the second graph.
`method`	A character string specifying the similarity method: "frobenius" (default) or "cosine".

Value

A numeric value between 0 (no similarity) and 1 (identical graphs).

Examples

```
library(igraph)
g1 <- graph_from_edgelist(matrix(c(1, 2, 2, 3), ncol = 2, byrow = TRUE), directed = FALSE)
g2 <- graph_from_edgelist(matrix(c(1, 2, 2, 4), ncol = 2, byrow = TRUE), directed = FALSE)
adjacency_similarity(g1, g2, method = "frobenius") # Output: 0.5
adjacency_similarity(g1, g2, method = "cosine") # Output: 0.5
```

---

anno_edge *Use dataframe to annotate edges of an igraph*

Description

Use dataframe to annotate edges of an igraph

Usage

```
anno_edge(go, anno_tab, verbose = TRUE)
```

Arguments

go	metanet an igraph object
anno_tab	a dataframe using to annotate (with rowname or a name column)
verbose	logical

Value

a annotated igraph object

See Also

Other manipulate: anno_vertex(), c_net_annotate(), c_net_filter(), c_net_load(), c_net_save(), c_net_union(), get_e(), get_n(), get_v(), is_metanet()

Examples

```
data("c_net")
anno <- data.frame("from" = "s__Pelomonas_puraquae", "to" = "s__un_g__Rhizobium", new_atr = "new")
anno_edge(co_net, anno) -> anno_net
```

anno_vertex *Use data.frame to annotate vertexes of metanet*

Description

Use data.frame to annotate vertexes of metanet

Usage

```
anno_vertex(go, anno_tab, verbose = TRUE)
```

Arguments

go	metanet object
anno_tab	a dataframe using to annotate (with rowname or a "name" column)
verbose	logical

Value

a annotated metanet object

See Also

Other manipulate: anno_edge(), c_net_annotate(), c_net_filter(), c_net_load(), c_net_save(), c_net_union(), get_e(), get_n(), get_v(), is_metanet()

Examples

```
data("c_net")
data("otutab", package = "pcutils")
anno_vertex(co_net, taxonomy)
```

arc_count *Edgelist*

Description

Edgelist for c_net_from_edgelist()

arc_taxonomy *Edgelist*

Description

Edgelist for c_net_from_edgelist()

as.ggig *Transfer an igraph object to a ggig*

Description

Transfer an igraph object to a ggig

Usage

```
as.ggig(go, coors = NULL)
```

Arguments

go	igraph or meatnet
coors	coordinates for nodes,columns: name, X, Y

Value

ggig object

See Also

Other plot: c_net_plot(), input_cytoscape(), input_gephi(), netD3plot(), olympic_rings_net(), plot.ggig(), twocol_edgelist(), venn_net()

Examples

```
as.ggig(co_net, coors = c_net_layout(co_net)) -> ggig
plot(ggig)
as.ggig(multi1, coors = c_net_layout(multi1)) -> ggig
plot(ggig, labels_num = 0.3)
```

as_arc *Layout as a arc*

Description

Layout as a arc

Usage

```
as_arc(angle = 0, arc = pi)
```

Arguments

angle	anticlockwise rotation angle
arc	the radian of arc

Value

A two-column matrix, each row giving the coordinates of a vertex, according to the ids of the vertex ids.

See Also

Other layout: as_circle_tree(), as_line(), as_multi_layer(), as_poly_sector(), as_polyarc(), as_polycircle(), as_polygon(), c_net_layout()

Examples

```
as_arc()(co_net)
c_net_plot(co_net, coors = as_arc(pi / 2))
```

---

as_circle_tree *Layout as a circle_tree*

Description

Layout as a circle_tree

Usage

```
as_circle_tree()
```

Value

A two-column matrix, each row giving the coordinates of a vertex, according to the ids of the vertex ids.

See Also

Other layout: as_arc(), as_line(), as_multi_layer(), as_poly_sector(), as_polyarc(), as_polycircle(), as_polygon(), c_net_layout()

as_coors *Transfer to a coors object*

Description

Transfer to a coors object

Usage

```
as_coors(coors, curved = NULL)
```

Arguments

coors	data.frame
curved	line curved

Value

coors object

as_line *Layout as a line*

Description

Layout as a line

Usage

```
as_line(angle = 0)
```

Arguments

angle	anticlockwise rotation angle

Value

A two-column matrix, each row giving the coordinates of a vertex, according to the ids of the vertex ids.

See Also

Other layout: as_arc(), as_circle_tree(), as_multi_layer(), as_poly_sector(), as_polyarc(), as_polycircle(), as_polygon(), c_net_layout()

Examples

```
as_line()(co_net)
c_net_plot(co_net, coors = as_line(pi / 2))
```

`as_multi_layer` *Layout as a multi_layer*

Description

Layout as a multi_layer

Usage

```
as_multi_layer(n = 3, layout = on_grid())
```

Arguments

`n`	how many arcs of this multi_layer
`layout`	see method in `c_net_layout`

Value

A two-column matrix, each row giving the coordinates of a vertex, according to the ids of the vertex ids.

See Also

Other layout: `as_arc()`, `as_circle_tree()`, `as_line()`, `as_poly_sector()`, `as_polyarc()`, `as_polycircle()`, `as_polygon()`, `c_net_layout()`

Examples

```
as_multi_layer()(co_net)
```

as_polyarc *Layout as a polyarc*

Description

Layout as a polyarc

Usage

```
as_polyarc(n = 3, space = pi/3)
```

Arguments

n	how many arcs of this poly_arc
space	the space between each arc, default: pi/3

Value

A two-column matrix, each row giving the coordinates of a vertex, according to the ids of the vertex ids.

See Also

Other layout: as_arc(), as_circle_tree(), as_line(), as_multi_layer(), as_poly_sector(), as_polycircle(), as_polygon(), c_net_layout()

Examples

```
as_polyarc()(co_net)
```

as_polycircle *Layout as a polycircle*

Description

Layout as a polycircle

Usage

```
as_polycircle(n = 5)
```

Arguments

n	how many circles of this polycircle

Value

A two-column matrix, each row giving the coordinates of a vertex, according to the ids of the vertex ids.

See Also

Other layout: as_arc(), as_circle_tree(), as_line(), as_multi_layer(), as_poly_sector(), as_polyarc(), as_polygon(), c_net_layout()

Examples

```
as_polycircle()(co_net)
```

---

as_polygon *Layout as a polygon*

Description

Layout as a polygon

Usage

```
as_polygon(n = 3, line_curved = 0.5)
```

Arguments

n	how many edges of this polygon
line_curved	line_curved 0~0.5

Value

A two-column matrix, each row giving the coordinates of a vertex, according to the ids of the vertex ids.

See Also

Other layout: as_arc(), as_circle_tree(), as_line(), as_multi_layer(), as_poly_sector(), as_polyarc(), as_polycircle(), c_net_layout()

Examples

```
as_polygon()(co_net)
```

as_poly_sector *Layout as a multi_layer*

Description

Layout as a multi_layer

Usage

```
as_poly_sector(n = 3)
```

Arguments

n	how many arcs of this multi_layer

Value

A two-column matrix, each row giving the coordinates of a vertex, according to the ids of the vertex ids.

See Also

Other layout: as_arc(), as_circle_tree(), as_line(), as_multi_layer(), as_polyarc(), as_polycircle(), as_polygon(), c_net_layout()

cal_sim *Calculate similarity for one t(otutab)*

Description

Calculate similarity for one t(otutab)

Usage

```
cal_sim(totu, totu2 = NULL, method = "bray")
```

Arguments

totu	t(otutab), row are samples, column are features.
totu2	t(otutab) or NULL, row are samples, column are features.
method	Dissimilarity index, see vegdist.

Value

similarity = 1-distance

See Also

vegdist

Other calculate: c_net_calculate(), fast_cor(), p.adjust.table(), read_corr()

Examples

```
if (requireNamespace("vegan")) {
  data("otutab", package = "pcutils")
  t(otutab) -> totu
  cal_sim(totu) -> sim_corr
}
```

---

check_tabs *Check tables and extract common samples*

Description

Check tables and extract common samples

Usage

```
check_tabs(...)
```

Arguments

...	tables

Value

formatted tables

Examples

```
data("otutab", package = "pcutils")
check_tabs(otutab)
```

clean_igraph *Clean a igraph object*

Description

Clean a igraph object

Usage

```
clean_igraph(go, direct = NULL)
```

Arguments

go	igraph, metanet objects
direct	direct?

Value

a igraph object

clean_multi_edge_metanet *Clean multi edge metanet to plot*

Description

Clean multi edge metanet to plot

Usage

```
clean_multi_edge_metanet(go)
```

Arguments

go	metanet object

Value

metanet object

Examples

```
g <- igraph::make_ring(2)
g <- igraph::add_edges(g, c(1, 1, 1, 1, 2, 1))
plot(g)
plot(clean_multi_edge_metanet(g))
```

Cohesion *Cohesion calculation*

Description

Cohesion calculation

Plot cohesion

Usage

```
Cohesion(otutab, reps = 200, threads = 1, mycor = NULL, verbose = TRUE)

## S3 method for class 'cohesion'
plot(x, group, metadata, mode = 1, ...)
```

Arguments

otutab	otutab
reps	iteration time
threads	threads
mycor	a correlation matrix you want to use, skip the null model build when mycor is not NULL, default: NULL
verbose	verbose
x	Cohesion() result (cohesion object)
group	group name in colnames(metadata)
metadata	metadata
mode	plot mode, 1~2
...	additional arguments for group_box (mode=1) or group_box (mode=2)

Value

Cohesion object: a list with two dataframe

a ggplot

References

Herren, C. M. & McMahon, K. (2017) Cohesion: a method for quantifying the connectivity of microbial communities. doi:10.1038/ismej.2017.91.

Examples

```
data("otutab", package = "pcutils")
# set reps at least 99 when you run.
Cohesion(otutab[1:50, ], reps = 19) -> cohesion_res
if (requireNamespace("ggpubr")) {
  plot(cohesion_res, group = "Group", metadata = metadata, mode = 1)
  plot(cohesion_res, group = "Group", metadata = metadata, mode = 2)
}
```

`co_net` *MetaNet networks*

Description

MetaNet co_nets

`co_net2` *MetaNet networks*

Description

MetaNet co_nets

`co_net_rmt` *MetaNet networks*

Description

MetaNet co_nets

c_net_annotate *Annotate a metanet*

Description

Annotate a metanet

Usage

```
c_net_annotate(go, anno_tab, mode = "v", verbose = TRUE)
```

Arguments

go	metanet object
anno_tab	a dataframe using to annotate (mode v, e), or a list (mode n)
mode	"v" for vertex, "e" for edge, "n" for network
verbose	logical

Value

a annotated metanet object

See Also

Other manipulate: anno_edge(), anno_vertex(), c_net_filter(), c_net_load(), c_net_save(), c_net_union(), get_e(), get_n(), get_v(), is_metanet()

Examples

```
data("c_net")
anno <- data.frame("name" = "s__Pelomonas_puraquae", new_atr = "new")
co_net_new <- c_net_annotate(co_net, anno, mode = "v")
get_v(co_net_new, c("name", "new_atr"))

anno <- data.frame("from" = "s__Pelomonas_puraquae", "to" = "s__un_g__Rhizobium", new_atr = "new")
co_net_new <- c_net_annotate(co_net, anno, mode = "e")
get_e(co_net_new, c("from", "to", "new_atr"))

co_net_new <- c_net_annotate(co_net, list(new_atr = "new"), mode = "n")
get_n(co_net_new)
```

c_net_build *Construct a metanet from a corr object*

Description

Construct a metanet from a corr object

Usage

```
c_net_build(
  corr,
  r_threshold = 0.6,
  p_threshold = 0.05,
  use_p_adj = TRUE,
  delete_single = TRUE
)
```

Arguments

corr	corr object from `c_net_calculate()` or `read_corr()`.
r_threshold	r_threshold (default: >0.6).
p_threshold	p_threshold (default: <0.05).
use_p_adj	use the p.adjust instead of p.value (default: TRUE), if p.adjust not in the corr object, use p.value.
delete_single	should delete single vertexes?

Value

an metanet object

See Also

Other build: `c_net_from_edgelist()`, `c_net_set()`, `c_net_update()`, `multi_net_build()`

Examples

```
data("otutab", package = "pcutils")
t(otutab) -> totu
metadata[, 3:10] -> env
c_net_calculate(totu) -> corr
c_net_build(corr, r_threshold = 0.65) -> co_net

c_net_calculate(totu, env) -> corr2
c_net_build(corr2) -> co_net2
```

`c_net_calculate` *Calculate correlation for one or two t(otutab), or distance for one t(otutab).*

Description

Calculate correlation for one or two t(otutab), or distance for one t(otutab).

Usage

```
c_net_calculate(
  totu,
  totu2 = NULL,
  method = "spearman",
  filename = FALSE,
  p.adjust.method = NULL,
  p.adjust.mode = "all",
  threads = 1,
  verbose = TRUE
)
```

Arguments

`totu`	t(otutab), row are samples, column are features.
`totu2`	t(otutab2) or NULL, row are samples, column are features.
`method`	"spearman" (default), "pearson", "sparcc", or distance index from `vegdist`.
`filename`	the prefix of saved .corr file or FALSE.
`p.adjust.method`	see `p.adjust`
`p.adjust.mode`	see `p.adjust.table`
`threads`	threads, default: 1.
`verbose`	verbose, default: TRUE.

Value

a corr object with 3 elements:

`r`	default: spearman correlation
`p.value`	default: p-value of spearman correlation
`p.adjust`	default p.adjust.method = NULL

See Also

Other calculate: `cal_sim()`, `fast_cor()`, `p.adjust.table()`, `read_corr()`

Examples

```
data("otutab", package = "pcutils")
t(otutab) -> totu
c_net_calculate(totu) -> corr
metadata[, 3:10] -> env
c_net_calculate(totu, env) -> corr2
```

`c_net_compare` *Compare Two Networks*

Description

Compare Two Networks

Usage

```
c_net_compare(g1, g2)
```

Arguments

g1	network1
g2	network2

Value

A list containing the following elements:

- `g1`: The first network.
- `g2`: The second network.
- `g_union`: The union of the two networks.
- `g_inter`: The intersection of the two networks.
- `net_par_df`: A data frame containing the network parameters.
- `net_similarity`: A list containing the similarity metrics.

Examples

```
data("c_net")
set.seed(12)
co_net_p1 <- c_net_filter(co_net, name %in% sample(V(co_net)$name, 300))
co_net_p2 <- c_net_filter(co_net, name %in% sample(V(co_net)$name, 300))
c_net_compare(co_net_p1, co_net_p2) -> c_net_comp
plot(c_net_comp)
```

c_net_difference *Difference two networks*

Description

Difference two networks

Usage

```
c_net_difference(go1, go2, ...)
```

Arguments

go1	metanet object
go2	metanet object
...	add

Value

metanet

c_net_ego *Extract ego-centric subnetwork with preserved class attributes*

Description

Wrapper around `igraph::make_ego_graph()` that ensures output retains "metanet" and "igraph" class structure. Supports single or multiple center nodes.

Usage

```
c_net_ego(graph, nodes, order = 1, mode = "all")
```

Arguments

graph	An igraph object with potential "metanet" class
nodes	Center node(s) for subnetwork extraction (vertex IDs or names)
order	Integer specifying the order of neighbors to include
mode	Character scalar, either "in", "out" or "all" for directed networks

Value

metanet

Examples

```
library(igraph)
c_net_plot(co_net)
c_net_plot(c_net_ego(co_net, "s__Kribbella_catacumbae"))
nodes <- c("s__Kribbella_catacumbae", "s__Verrucosispora_andamanensis")
c_net_plot(c_net_ego(co_net, nodes))
```

`c_net_filter` *Filter a network according to some attributes*

Description

Filter a network according to some attributes

Usage

```
c_net_filter(go, ..., mode = "v")
```

Arguments

go	metanet object
...	some attributes of vertex and edge
mode	"v" or "e"

Value

metanet

See Also

Other manipulate: `anno_edge()`, `anno_vertex()`, `c_net_annotate()`, `c_net_load()`, `c_net_save()`, `c_net_union()`, `get_e()`, `get_n()`, `get_v()`, `is_metanet()`

Examples

```
data("multi_net")
c_net_filter(multi1, v_group %in% c("omic1", "omic2"))
```

`c_net_from_edgelist` *Construct a network from edge_list dataframe*

Description

Construct a network from edge_list dataframe

Usage

```
c_net_from_edgelist(
  edgelist,
  vertex_df = NULL,
  direct = FALSE,
  e_type = NULL,
  e_class = NULL
)
```

Arguments

`edgelist`	first is source, second is target, others are annotation
`vertex_df`	vertex metadata data.frame
`direct`	logical
`e_type`	set e_type
`e_class`	set e_class

Value

metanet

See Also

Other build: `c_net_build()`, `c_net_set()`, `c_net_update()`, `multi_net_build()`

Examples

```
data(edgelist)
edge_net <- c_net_from_edgelist(arc_count, vertex_df = arc_taxonomy)
edge_net <- c_net_set(edge_net, vertex_class = "Phylum", edge_width = "n")
c_net_plot(edge_net)
```

c_net_highlight *Highlight specific nodes in a network*

Description

Adds highlight markers to specified nodes and grays out non-highlighted nodes. Preserves all existing vertex/edge attributes and class structure.

Usage

```
c_net_highlight(graph, nodes = NULL, edges = NULL, gray_color = "gray80")
```

Arguments

graph	An igraph/metatnet object
nodes	Vector of node names to highlight
edges	a data.frame of edges to highlight, colnames must be "from" and "to"
gray_color	Color for non-highlighted nodes (default: "gray80")

Value

metanet

Examples

```
par(mfrow = c(1, 3))
nodes <- c("s__Kribbella_catacumbae", "s__Verrucosispora_andamanensis")
nodes <- V(c_net_ego(co_net, nodes))$name
g_hl <- c_net_highlight(co_net, nodes = nodes)
plot(g_hl) # Highlighted nodes keep colors, others turn gray
get_e(co_net) %>% head(20) -> hl_edges
g_hl2 <- c_net_highlight(co_net, edges = hl_edges[, 2:3])
c_net_plot(g_hl2)
g_hl3 <- c_net_highlight(co_net, nodes = nodes, edges = hl_edges[, 2:3])
c_net_plot(g_hl3)
```

c_net_intersect *Intersect two networks*

Description

Intersect two networks

Usage

```
c_net_intersect(go1, go2, ...)
```

Arguments

go1	metanet object
go2	metanet object
...	add

Value

metanet

`c_net_layout` *Layout coordinates*

Description

Layout coordinates

Usage

```
c_net_layout(
  go,
  method = igraph::nicely(),
  order_by = NULL,
  order_ls = NULL,
  seed = 1234,
  line_curved = 0.5,
  rescale = TRUE,
  ...
)
```

Arguments

go	igraph or metanet
method	(1) as_line(), as_arc(), as_polygon(), as_polyarc(), as_polycircle(), as_circle_tree(); (2) as_star(), as_tree(), in_circle(), nicely(), on_grid(), on_sphere(),randomly(), with_dh(), with_fr(), with_gem(), with_graphopt(), with_kk(),with_lgl(), with_mds(),. see layout_; (3) a character, "auto","backbone","centrality","circlepack","dendrogram", "eigen","focus","hive","igraph","linear","manual","matrix", "partition","pmds","stress","treemap","unroo see create_layout
order_by	order nodes according to a node attribute
order_ls	manual the discrete variable with a vector, or continuous variable with "desc" to decreasing
seed	random seed
line_curved	consider line curved, only for some layout methods like as_line(), as_polygon().default:0
rescale	logical, scale the X, Y to (-1,1)
...	add

Value

coors object: coordinates for nodes, columns: name, X, Y; curved for edges, columns: from, to, curved;

See Also

Other layout: as_arc(), as_circle_tree(), as_line(), as_multi_layer(), as_poly_sector(), as_polyarc(), as_polycircle(), as_polygon()

Examples

```
library(igraph)
c_net_layout(co_net) -> coors
c_net_plot(co_net, coors)
c_net_plot(co_net, c_net_layout(co_net, in_circle()), vertex.size = 2)
c_net_plot(co_net, c_net_layout(co_net, in_circle(), order_by = "v_class"), vertex.size = 2)
c_net_plot(co_net, c_net_layout(co_net, in_circle(), order_by = "size", order_ls = "desc"))
c_net_plot(co_net, c_net_layout(co_net, as_polygon(3)))
```

---

c_net_load *Load network file*

Description

Load network file

Usage

```
c_net_load(filename, format = "data.frame")
```

Arguments

filename	filename
format	"data.frame","graphml"

Value

metanet

See Also

Other manipulate: anno_edge(), anno_vertex(), c_net_annotate(), c_net_filter(), c_net_save(), c_net_union(), get_e(), get_n(), get_v(), is_metanet()

c_net_plot *Plot a metanet*

Description

Plot a metanet

Usage

```
c_net_plot(
  go,
  coors = NULL,
  ...,
  labels_num = NULL,
  vertex_size_range = NULL,
  edge_width_range = NULL,
  plot_module = FALSE,
  mark_module = FALSE,
  mark_color = NULL,
  mark_alpha = 0.3,
  module_label = FALSE,
  module_label_cex = 2,
  module_label_color = "black",
  module_label_just = c(0.5, 0.5),
  pie_value = NULL,
  pie_color = NULL,
  legend = TRUE,
  legend_number = FALSE,
  legend_cex = 1,
 legend_position = c(left_leg_x = -2, left_leg_y = 1, right_leg_x = 1.2, right_leg_y =
    1),
  group_legend_title = NULL,
  group_legend_order = NULL,
  color_legend = TRUE,
  color_legend_order = NULL,
  size_legend = FALSE,
  size_legend_title = "Node Size",
  edge_legend = TRUE,
  edge_legend_title = "Edge type",
  edge_legend_order = NULL,
  width_legend = FALSE,
  width_legend_title = "Edge width",
  lty_legend = FALSE,
  lty_legend_title = "Edge class",
  lty_legend_order = NULL,
  module_legend = FALSE,
  module_legend_title = "Module",
```

```
  module_legend_order = NULL,
  pie_legend = FALSE,
  pie_legend_title = "Pie part",
  pie_legend_order = NULL,
  label_cex = 1,
  arrow_size_cex = 1,
  arrow_width_cex = 1,
  params_list = NULL,
  rescale = FALSE,
  seed = 1234
)
```

Arguments

go	an igraph or metanet object
coors	the coordinates you saved
...	additional parameters for igraph.plotting
labels_num	show how many labels, >1 indicates number, <1 indicates fraction, "all" indicates all.
vertex_size_range	the vertex size range, e.g. c(1,10)
edge_width_range	the edge width range, e.g. c(1,10)
plot_module	logical, plot module?
mark_module	logical, mark the modules?
mark_color	mark color
mark_alpha	mark fill alpha, default 0.3
module_label	show module label?
module_label_cex	module label cex
module_label_color	module label color
module_label_just	module label just, default c(0.5,0.5)
pie_value	a dataframe using to plot pie (with rowname or a "name" column)
pie_color	color vector
legend	all legends
legend_number	legend with numbers
legend_cex	character expansion factor relative to current par("cex"), default: 1
legend_position	legend_position, default: c(left_leg_x=-1.9,left_leg_y=1,right_leg_x=1.2,right_leg_y=1)
group_legend_title	group_legend_title, length must same to the numbers of v_group

group_legend_order
: group_legend_order vector

color_legend
: logical

color_legend_order
: color_legend_order vector

size_legend
: logical

size_legend_title
: size_legend_title

edge_legend
: logical

edge_legend_title
: edge_legend_title

edge_legend_order
: edge_legend_order vector, e.g. c("positive","negative")

width_legend
: logical

width_legend_title
: width_legend_title

lty_legend
: logical

lty_legend_title
: lty_legend_title

lty_legend_order
: lty_legend_order

module_legend
: logical

module_legend_title
: module_legend_title

module_legend_order
: module_legend_order

pie_legend
: logical

pie_legend_title
: pie_legend_title

pie_legend_order
: pie_legend_order

label_cex
: label cex, default 1, relative to the vertex size

arrow_size_cex
: arrow size cex, default 1, relative to the vertex size

arrow_width_cex
: arrow width cex, default 1, relative to the vertex size

params_list
: a list of parameters, e.g. list(edge_legend = TRUE, lty_legend = FALSE), when the parameter is duplicated, the format argument will be used rather than the argument in params_list.

rescale
: Logical constant, whether to rescale the coordinates to the (-1,1).

seed
: random seed, default:1234, make sure each plot is the same.

Value

a network plot

See Also

Other plot: as.ggig(), input_cytoscape(), input_gephi(), netD3plot(), olympic_rings_net(), plot.ggig(), twocol_edgelist(), venn_net()

Examples

```
data("c_net")
c_net_plot(co_net)
c_net_plot(co_net2)
c_net_plot(multi1)
```

---

c_net_save *Save network file*

Description

Save network file

Usage

```
c_net_save(go, filename = "net", format = "data.frame")
```

Arguments

go	metanet network
filename	filename
format	"data.frame","graphml"

Value

No value

See Also

Other manipulate: anno_edge(), anno_vertex(), c_net_annotate(), c_net_filter(), c_net_load(), c_net_union(), get_e(), get_n(), get_v(), is_metanet()

c_net_set *Set basic attributes from totu table*

Description

Set basic attributes from totu table

Usage

```
c_net_set(
  go,
  ...,
  vertex_group = "v_group",
  vertex_class = "v_class",
  vertex_size = "size",
  edge_type = "e_type",
  edge_class = "e_class",
  edge_width = "width",
  node_break = 5,
  edge_break = 5,
  initialize = TRUE
)
```

Arguments

go	metanet an igraph object
...	some data.frames to annotate go
vertex_group	choose which column to be vertex_group (map to vertex_shape)
vertex_class	choose which column to be vertex_class (map to vertex_color)
vertex_size	choose which column to be vertex_size (map to vertex_size)
edge_type	choose which column to be edge_type (map to edge_color)
edge_class	choose which column to be edge_class (map to edge_linetype)
edge_width	choose which column to be edge_width (map to edge_width)
node_break	node_break if v_class is numeric, default: 5
edge_break	edge_break if e_type is numeric, default: 5
initialize	initialize, default: TRUE

Value

a metanet object

See Also

Other build: c_net_build(), c_net_from_edgelist(), c_net_update(), multi_net_build()

Examples

```
data("otutab", package = "pcutils")
t(otutab) -> totu
metadata[, 3:10] -> env

data("c_net")
co_net <- c_net_set(co_net, taxonomy, data.frame("Abundance" = colSums(totu)),
  vertex_class = "Phylum", vertex_size = "Abundance"
)
co_net2 <- c_net_set(co_net2, taxonomy, data.frame(name = colnames(env), env = colnames(env)),
  vertex_class = c("Phylum", "env")
)
co_net2 <- c_net_set(co_net2, data.frame("Abundance" = colSums(totu)), vertex_size = "Abundance")
```

---

c_net_stability *Evaluate the stability of a network*

Description

$$Vi = \frac{E - Ei}{E}$$

E is the global efficiency and Ei is the global efficiency after the removal of the node i and its entire links.

Usage

```
c_net_stability(
  go_ls,
  mode = "robust_test",
  partial = 0.5,
  step = 10,
  reps = 9,
  threads = 1,
  verbose = TRUE,
  keystone = FALSE
)

robust_test(
  go_ls,
  partial = 0.5,
  step = 10,
  reps = 9,
  threads = 1,
  verbose = TRUE
)
```

```
vulnerability(go_ls, threads = 1, verbose = TRUE)

robustness(go_ls, keystone = FALSE, reps = 9, threads = 1, verbose = TRUE)
```

Arguments

go_ls	an igraph object or igraph list.
mode	"robust_test", "vulnerability", "robustness"
partial	how much percent vertexes be removed in total (default: 0.5, only for robust_test)
step	how many nodes be removed each time? (default: 10, only for robust_test)
reps	simulation number (default: 9)
threads	threads
verbose	verbose
keystone	remove 70%% keystones instead of remove 50%% nodes (default: False, only for robustness)

Value

a data.frame

data.frame (robustness class)

a vector

Examples

```
data("c_net")
if (requireNamespace("ggpmisc")) {
  c_net_stability(co_net, mode = "robust_test", step = 20, reps = 9) -> robust_res
  plot(robust_res, index = "Average_degree", mode = 2)
}

c_net_stability(co_net, mode = "vulnerability") -> vulnerability_res
plot(vulnerability_res)

robustness(co_net) -> robustness_res
plot(robustness_res)

module_detect(co_net) -> co_net_modu
zp_analyse(co_net_modu, mode = 2) -> co_net_modu

c_net_stability(co_net_modu, mode = "robustness", keystone = TRUE) -> robustness_res
plot(robustness_res)
```

c_net_union *Union two networks*

Description

Union two networks

Usage

```
c_net_union(go1, go2, ...)
```

Arguments

go1	metanet object
go2	metanet object
...	add

Value

metanet

See Also

Other manipulate: anno_edge(), anno_vertex(), c_net_annotate(), c_net_filter(), c_net_load(), c_net_save(), get_e(), get_n(), get_v(), is_metanet()

Examples

```
g1 <- make_graph(edges = c("1", 2, 2, 3, 3, 4, 4, 5, 5, 1), directed = FALSE) %>% as.metanet()
g2 <- make_graph(edges = c("4", 5, 5, 6, 6, 7, 7, 8, 8, 4), directed = FALSE) %>% as.metanet()
par(mfrow = c(1, 3))
plot(c_net_union(g1, g2))
plot(c_net_intersect(g1, g2))
plot(c_net_difference(g1, g2))
```

c_net_update *Update a metanet object or transform igraph object to metanet object*

Description

Update a metanet object or transform igraph object to metanet object

Usage

```
c_net_update(
  go,
  node_break = 5,
  edge_break = 5,
  initialize = FALSE,
  verbose = TRUE,
  uniq_v_class = FALSE
)
```

Arguments

go	a metanet object or igraph object
node_break	node_break if v_class is numeric, default: 5
edge_break	edge_break if e_type is numeric, default: 5
initialize	initialize?
verbose	verbose?
uniq_v_class	if TRUE, add prefix to v_class if multiple v_class belong to same v_group.

Value

metanet

See Also

Other build: c_net_build(), c_net_from_edgelist(), c_net_set(), multi_net_build()

---

df2net_tree *Transform a dataframe to a network edgelist.*

Description

Transform a dataframe to a network edgelist.

Usage

```
df2net_tree(test, fun = sum)
```

Arguments

test	df
fun	default: sum

Value

metanet

Examples

```
data("otutab", package = "pcutils")
cbind(taxonomy, num = rowSums(otutab))[1:20, ] -> test
df2net_tree(test) -> ttt
plot(ttt)
if (requireNamespace("ggraph")) plot(ttt, coors = as_circle_tree())
```

`extract_sample_net` *Extract each sample network from the whole network*

Description

Extract each sample network from the whole network

Usage

```
extract_sample_net(
  whole_net,
  otutab,
  threads = 1,
  save_net = FALSE,
  fast = TRUE,
  remove_negative = FALSE,
  verbose = TRUE
)
```

Arguments

`whole_net`	the whole network
`otutab`	otutab, columns are samples, these columns will be extract
`threads`	threads, default: 1
`save_net`	should save these sub_nets? FALSE or a filename
`fast`	less indexes for faster calculate ?
`remove_negative`	remove negative edge or not? default: FALSE
`verbose`	verbose

Value

a dataframe contains all sub_net parameters

See Also

Other topological: `fit_power()`, `get_group_skeleton()`, `links_stat()`, `nc()`, `net_par()`, `rand_net()`, `rand_net_par()`, `smallworldness()`

Examples

```
data(otutab, package = "pcutils")
extract_sample_net(co_net, otutab) -> sub_net_pars
```

`fast_cor` *Fast correlation calculation*

Description

Fast correlation calculation

Usage

```
fast_cor(totu, totu2 = NULL, method = c("pearson", "spearman"))
```

Arguments

`totu`	t(otutab), row are samples, column are features.
`totu2`	t(otutab) or NULL, row are samples, column are features.
`method`	"spearman" or "pearson"

Value

a list with 2 elements:

`r`	default: spearman correlation
`p.value`	default: p-value of spearman correlation

See Also

Other calculate: `c_net_calculate()`, `cal_sim()`, `p.adjust.table()`, `read_corr()`

Examples

```
data("otutab", package = "pcutils")
t(otutab[1:100, ]) -> totu
fast_cor(totu, method = "spearman") -> corr
```

`filter_n_module` *Filter some modules as others*

Description

Filter some modules as others

Combine or cut modules to module_number

Plot module tree

Usage

```
filter_n_module(go_m, n_node_in_module = 0, keep_id = NULL, delete = FALSE)

combine_n_module(go_m, module_number = 5)

plot_module_tree(go_m, module = "module", community = NULL, label.size = 2)
```

Arguments

`go_m`	module metanet
`n_node_in_module`	transfer the modules less than n_node_in_module to "others"
`keep_id`	keep modules ids, will not be "others"
`delete`	logical, delete others modules? default:FALSE, the others module will be "others".
`module_number`	number of modules
`module`	which column name is module. default: "module"
`community`	community object, default: NULL, use the community of go_m
`label.size`	label.size

Value

metanet with modules

ggplot

See Also

Other module: `get_community()`, `get_module()`, `get_module_eigen()`, `module_detect()`, `module_eigen()`, `module_net()`, `summary_module()`, `to_module_net()`, `zp_analyse()`

Examples

```
data("c_net")
module_detect(co_net) -> co_net_modu
filter_n_module(co_net_modu, n_node_in_module = 30) -> co_net_modu
if (requireNamespace("ggtree") && requireNamespace("treeio")) plot_module_tree(co_net_modu)
combine_n_module(co_net_modu, 20) -> co_net_modu1
if (requireNamespace("ggtree") && requireNamespace("treeio")) plot_module_tree(co_net_modu1)
```

`fit_power` *Fit power-law distribution for an igraph*

Description

Fit power-law distribution for an igraph

Usage

```
fit_power(go, p.value = FALSE)
```

Arguments

go	igraph
p.value	calculate p.value

Value

ggplot

See Also

Other topological: `extract_sample_net()`, `get_group_skeleton()`, `links_stat()`, `nc()`, `net_par()`, `rand_net()`, `rand_net_par()`, `smallworldness()`

Examples

```
fit_power(co_net)
```

get_community *Get community*

Description

Get community

Usage

```
get_community(go_m)
```

Arguments

go_m	module metanet

Value

community

See Also

Other module: filter_n_module(), get_module(), get_module_eigen(), module_detect(), module_eigen(), module_net(), summary_module(), to_module_net(), zp_analyse()

get_e *Get edge information*

Description

Get edge information

Usage

```
get_e(go, index = NULL)
```

Arguments

go	metanet object
index	attribute name, default: NULL

Value

data.frame

See Also

Other manipulate: anno_edge(), anno_vertex(), c_net_annotate(), c_net_filter(), c_net_load(), c_net_save(), c_net_union(), get_n(), get_v(), is_metanet()

`get_group_skeleton` *Get skeleton network according to a group*

Description

Get skeleton network according to a group

Skeleton plot

Usage

```
get_group_skeleton(go, Group = "v_class", count = NULL, top_N = 8)

skeleton_plot(ske_net, split_e_type = TRUE, ...)
```

Arguments

`go`	network
`Group`	vertex column name
`count`	take which column count, default: NULL
`top_N`	top_N
`ske_net`	skeleton
`split_e_type`	split by e_type? default: TRUE
`...`	additional parameters for `igraph.plotting`

Value

skeleton network

See Also

Other topological: `extract_sample_net()`, `fit_power()`, `links_stat()`, `nc()`, `net_par()`, `rand_net()`, `rand_net_par()`, `smallworldness()`

Examples

```
get_group_skeleton(co_net) -> ske_net
skeleton_plot(ske_net)
```

get_module *Get module*

Description

Get module

Usage

```
get_module(go_m)
```

Arguments

go_m	module metanet

Value

module

See Also

Other module: filter_n_module(), get_community(), get_module_eigen(), module_detect(), module_eigen(), module_net(), summary_module(), to_module_net(), zp_analyse()

get_module_eigen *Get module_eigen*

Description

Get module_eigen

Usage

```
get_module_eigen(go_m)
```

Arguments

go_m	module metanet

Value

module_eigen

See Also

Other module: filter_n_module(), get_community(), get_module(), module_detect(), module_eigen(), module_net(), summary_module(), to_module_net(), zp_analyse()

get_n *Get network information*

Description

Get network information

Usage

```
get_n(go, index = NULL, simple = FALSE)
```

Arguments

go	metanet object
index	attribute name, default: NULL
simple	logical, get simple index

Value

data.frame

See Also

Other manipulate: anno_edge(), anno_vertex(), c_net_annotate(), c_net_filter(), c_net_load(), c_net_save(), c_net_union(), get_e(), get_v(), is_metanet()

get_v *Get vertex information*

Description

Get vertex information

Usage

```
get_v(go, index = NULL)
```

Arguments

go	metanet object
index	attribute name, default: NULL

Value

data.frame

See Also

Other manipulate: anno_edge(), anno_vertex(), c_net_annotate(), c_net_filter(), c_net_load(), c_net_save(), c_net_union(), get_e(), get_n(), is_metanet()

---

g_layout *Layout with group*

Description

Layout with group

Usage

```
g_layout(
  go,
  group = "module",
  group_order = NULL,
  layout1 = in_circle(),
  zoom1 = 20,
  layout2 = in_circle(),
  zoom2 = 3,
  show_big_layout = FALSE,
  rescale = TRUE,
  ...
)
```

Arguments

go	igraph or metanet object
group	group name (default: module)
group_order	group_order
layout1	layout1 method, one of (1) a dataframe or matrix: rowname is group, two columns are X and Y (2) function: layout method for c_net_layout default: in_circle()
zoom1	big network layout size
layout2	one of functions: layout method for c_net_layout, or a list of functions.
zoom2	average sub_network layout size, or numeric vector, or "auto"
show_big_layout	show the big layout to help you adjust.
rescale	logical, scale the X, Y to (-1,1)
...	add

Value

coors

See Also

Other g_layout: g_layout_nice(), g_layout_poly_sector(), g_layout_polygon()

Examples

```
data("c_net")
module_detect(co_net, method = "cluster_fast_greedy") -> co_net_modu
g_layout(co_net_modu, group = "module", zoom1 = 30, zoom2 = "auto", layout2 = as_line()) -> oridata
plot(co_net_modu, coors = oridata)
```

---

g_layout_nice *Layout with group nicely*

Description

Layout with group nicely

Usage

```
g_layout_nice(go, group = "module", mode = "circlepack", group_zoom = 1, ...)

get_big_lay_nice(go, group = "module", mode = "circlepack", ...)

g_layout_circlepack(go, group = "module", ...)

g_layout_treemap(go, group = "module", ...)

g_layout_backbone(go, group = "module", ...)

g_layout_stress(go, group = "module", ...)
```

Arguments

go	igraph or metanet
group	group name (default: module)
mode	circlepack, treemap, backbone, stress
group_zoom	zoom for each group
...	add

Value

coors

See Also

Other g_layout: g_layout(), g_layout_poly_sector(), g_layout_polygon()

Examples

```
data("c_net")
module_detect(co_net, method = "cluster_fast_greedy") -> co_net_modu
if (requireNamespace("ggraph")) {
  plot(co_net_modu, coors = g_layout_nice(co_net_modu, group = "module"))
 plot(co_net_modu, coors = g_layout_nice(co_net_modu, group = "module", mode = "treemap"))
}
```

---

g_layout_polygon *Layout with group as a polygon*

Description

Layout with group as a polygon

Layout with group as a polyarc

Layout with group as a polycircle

Layout with group as a multi_layer

Usage

```
g_layout_polygon(
  go,
  group = "v_group",
  group_order = NULL,
  group2 = NULL,
  group2_order = NULL,
  line_curved = 0.5
)

g_layout_polyarc(
  go,
  group = "v_group",
  group_order = NULL,
  group2 = NULL,
  group2_order = NULL,
  space = pi/4,
  scale_node_num = TRUE
)

g_layout_polycircle(
  go,
  group = "v_group",
  group_order = NULL,
  group2 = NULL,
  group2_order = NULL
```

```
)

g_layout_multi_layer(
  go,
  layout = igraph::in_circle(),
  group = "v_group",
  group_order = NULL,
  group2 = NULL,
  group2_order = NULL,
  scale_node_num = TRUE
)
```

Arguments

go	igraph
group	group name (default:v_group)
group_order	group_order
group2	group2 name, will order nodes in each group according to group2_order
group2_order	group2_order
line_curved	line_curved 0~1
space	the space between each arc, default: pi/4
scale_node_num	scale with the node number in each group
layout	see method in c_net_layout

Value

coors

See Also

Other g_layout: g_layout(), g_layout_nice(), g_layout_poly_sector()

Examples

```
g_layout_polygon(multi1) -> oridata
c_net_plot(multi1, oridata)
g_layout_polyarc(multi1, group2 = "v_class", group2_order = c(LETTERS[4:1])) -> oridata
c_net_plot(multi1, oridata)
g_layout_polycircle(co_net2, group2 = "v_class") -> oridata
c_net_plot(co_net2, oridata)
g_layout_multi_layer(co_net2, group2 = "v_class") -> oridata
c_net_plot(co_net2, oridata)
```

`g_layout_poly_sector` *Layout with group*

Description

Layout with group

Usage

```
g_layout_poly_sector(
  go,
  group = "v_group",
  group_order = NULL,
  group2 = NULL,
  group2_order = NULL,
  space = pi/4,
  jitter = 0,
  scale_node_num = TRUE,
  type = c("regular", "random"),
  mode = c("surface", "boundary"),
  curved = NULL
)
```

Arguments

`go`	igraph
`group`	group name (default:v_group)
`group_order`	group_order
`group2`	group2 name, will order nodes in each group according to group2_order
`group2_order`	group2_order
`space`	the space between each arc, default: pi/4
`jitter`	for surface-regular, defalut 0
`scale_node_num`	scale with the node number in each group
`type`	Type of distribution: "random", "regular"
`mode`	"surface", "boundary"
`curved`	Optional curved attribute for coors

Value

coors

See Also

Other g_layout: `g_layout()`, `g_layout_nice()`, `g_layout_polygon()`

input_cytoscape *Input a cyjs file exported by Cytoscape*

Description

Input a cyjs file exported by Cytoscape

Usage

```
input_cytoscape(file)
```

Arguments

file	cyjs file exported by Cytoscape

Value

list contains the igraph object and coordinates

See Also

Other plot: as.ggig(), c_net_plot(), input_gephi(), netD3plot(), olympic_rings_net(), plot.ggig(), twocol_edgelist(), venn_net()

input_gephi *Input a graphml file exported by Gephi*

Description

Input a graphml file exported by Gephi

Usage

```
input_gephi(file)
```

Arguments

file	graphml file exported by Gephi

Value

list contains the igraph object and coordinates

See Also

Other plot: as.ggig(), c_net_plot(), input_cytoscape(), netD3plot(), olympic_rings_net(), plot.ggig(), twocol_edgelist(), venn_net()

is_metanet *Is this object a metanet object?*

Description

Is this object a metanet object?

Usage

```
is_metanet(go)
```

Arguments

go	a test object

Value

logical

See Also

Other manipulate: anno_edge(), anno_vertex(), c_net_annotate(), c_net_filter(), c_net_load(), c_net_save(), c_net_union(), get_e(), get_n(), get_v()

Examples

```
data(c_net)
is_metanet(co_net)
```

links_stat *Link summary of the network*

Description

Link summary of the network

Usage

```
links_stat(
  go,
  group = "v_class",
  e_type = "all",
  topN = 10,
  colors = NULL,
  mode = 1,
  plot_param = list()
)
```

Arguments

go	igraph or metanet
group	summary which group of vertex attribution in names(vertex_attr(go))
e_type	"positive", "negative", "all"
topN	topN of group, default: 10
colors	colors
mode	1~2
plot_param	plot parameters

Value

plot

See Also

Other topological: `extract_sample_net()`, `fit_power()`, `get_group_skeleton()`, `nc()`, `net_par()`, `rand_net()`, `rand_net_par()`, `smallworldness()`

Examples

```
if (requireNamespace("circlize")) {
  links_stat(co_net, topN = 10)
  module_detect(co_net) -> co_net_modu
  links_stat(co_net_modu, group = "module")
}
if (requireNamespace("corrplot")) {
  links_stat(co_net, topN = 10, mode = 2)
}
```

---

metab *MetaNet networks abundance*

Description

MetaNet co_nets

---

metab_g *MetaNet networks metadata*

Description

MetaNet co_nets

micro *MetaNet networks abundance*

Description

MetaNet co_nets

micro_g *MetaNet networks metadata*

Description

MetaNet co_nets

module_detect *Detect the modules*

Description

Detect the modules

Usage

```
module_detect(
  go,
  method = "cluster_fast_greedy",
  n_node_in_module = 0,
  delete = FALSE
)
```

Arguments

go	an igraph object
method	cluster_method: "cluster_walktrap", "cluster_edge_betweenness", "cluster_fast_greedy", "cluster_spinglass"
n_node_in_module	transfer the modules less than n_node_in_module to "others"
delete	logical, delete others modules? default:FALSE, the others module will be "others".

Value

an igraph object

See Also

Other module: filter_n_module(), get_community(), get_module(), get_module_eigen(), module_eigen(), module_net(), summary_module(), to_module_net(), zp_analyse()

Examples

```
data("c_net")
module_detect(co_net) -> co_net_modu
```

---

module_eigen *Calculate the eigenvalue of each module and correlation of nodes and eigenvalue (node_eigen_cor).*

Description

Calculate the eigenvalue of each module and correlation of nodes and eigenvalue (node_eigen_cor).

Plot the expression of each modules

Usage

```
module_eigen(go_m, totu, cor_method = "spearman")

module_expression(
  go_m,
  totu,
  group = NULL,
  r_threshold = 0.6,
  x_order = NULL,
  facet_param = NULL,
  plot_eigen = FALSE
)
```

Arguments

go_m	module metanet
totu	original abundance table used for module_eigen().
cor_method	"pearson", "kendall", "spearman"
group	group variable for totu
r_threshold	the threshold for node_eigen_cor, default: 0.6.
x_order	order the x axis.
facet_param	parameters parse to facet_wrap, e.g. nrow=2.
plot_eigen	plot the eigen value line.

Value

module metanet with module_eigen

See Also

Other module: filter_n_module(), get_community(), get_module(), get_module_eigen(), module_detect(), module_net(), summary_module(), to_module_net(), zp_analyse()

Examples

```
data("otutab", package = "pcutils")
t(otutab) -> totu
data("c_net")
module_detect(co_net, n_node_in_module = 30) -> co_net_modu
module_eigen(co_net_modu, totu) -> co_net_modu
module_expression(co_net_modu, totu)
```

---

module_net *Generate a n-modules network*

Description

this is just a random generation method, the module number of result is not exactly the module_number, you can change the inter_module_density and intra_module_density to get the proper result.

Usage

```
module_net(
  module_number = 3,
  n_node_in_module = 30,
  intra_module_density = 0.3,
  inter_module_density = 0.01
)
```

Arguments

module_number	number of modules
n_node_in_module	number of nodes in each modules
intra_module_density	intra_module_density, recommend bigger than 20*inter_module_density, default:0.3
inter_module_density	inter_module_density, default:0.01

Value

n-modules metanet

See Also

Other module: filter_n_module(), get_community(), get_module(), get_module_eigen(), module_detect(), module_eigen(), summary_module(), to_module_net(), zp_analyse()

Examples

```
g1 <- module_net()
get_n(g1)
plot(g1, mark_module = TRUE)
plot(g1, coors = g_layout(g1, zoom2 = 15))
plot(g1, coors = g_layout_polyarc(g1, group = "module"))
plot(g1, coors = g_layout_polygon(g1, group = "module"))
```

---

multi1 *MetaNet networks*

Description

MetaNet co_nets

---

multi_net_build *Multi-omics network build*

Description

Multi-omics network build

Usage

```
multi_net_build(
  ...,
  mode = "full",
  method = "spearman",
  filename = FALSE,
  p.adjust.method = NULL,
  r_threshold = 0.6,
  p_threshold = 0.05,
  use_p_adj = TRUE,
  delete_single = TRUE
)
```

Arguments

...	some omics abundance tables
mode	"full"
method	"spearman" or "pearson"
filename	the prefix of saved .corr file or FALSE
p.adjust.method	see p.adjust
r_threshold	r_threshold (default: >0.6)
p_threshold	p_threshold (default: <0.05)
use_p_adj	use the p.adjust instead of p-value (default: TRUE)
delete_single	should delete single vertexes?

Value

metanet

See Also

Other build: c_net_build(), c_net_from_edgelist(), c_net_set(), c_net_update()

Examples

```
data("multi_test")
multi1 <- multi_net_build(list(Microbiome = micro, Metabolome = metab, Transcriptome = transc))
multi1 <- c_net_set(multi1, micro_g, metab_g, transc_g,
  vertex_class = c("Phylum", "kingdom", "type")
)
multi1 <- c_net_set(multi1, data.frame("Abundance1" = colSums(micro)),
  data.frame("Abundance2" = colSums(metab)), data.frame("Abundance3" = colSums(transc)),
  vertex_size = paste0("Abundance", 1:3)
)
c_net_plot(multi1)
```

---

nc *Calculate natural_connectivity*

Description

Calculate natural_connectivity

Usage

```
nc(p)
```

Arguments

p	an igraph or metanet object

Value

natural_connectivity (numeric)

References

`nc` in `ggClusterNet`

See Also

Other topological: extract_sample_net(), fit_power(), get_group_skeleton(), links_stat(), net_par(), rand_net(), rand_net_par(), smallworldness()

Examples

```
igraph::make_ring(10) %>% nc()
```

---

netD3plot *plot use networkD3*

Description

plot use networkD3

Usage

```
netD3plot(go, v_class = "v_class", ...)
```

Arguments

go	metanet
v_class	which attributes use to be v_class
...	see forceNetwork

Value

D3 plot

See Also

Other plot: as.ggig(), c_net_plot(), input_cytoscape(), input_gephi(), olympic_rings_net(), plot.ggig(), twocol_edgelist(), venn_net()

Examples

```
data("c_net")
plot(co_net2)
if (requireNamespace("networkD3")) {
  netD3plot(co_net2)
}
```

`net_par` *Calculate all topological indexes of a network*

Description

Calculate all topological indexes of a network

Add topological indexes for a network

Usage

```
net_par(
  go,
  mode = c("v", "e", "n", "all"),
  fast = TRUE,
  remove_negative = FALSE,
  only_topological = FALSE
)

c_net_index(go, force = FALSE)
```

Arguments

`go`	igraph or metanet
`mode`	calculate what? c("v", "e", "n", "all")
`fast`	less indexes for faster calculate ?
`remove_negative`	remove negative edge or not? default: FALSE
`only_topological`	only return topological indexes
`force`	replace existed net_par

Value

a 3-elements list

`n_index`	indexs of the whole network
`v_index`	indexs of each vertex
`e_index`	indexs of each edge

See Also

Other topological: extract_sample_net(), fit_power(), get_group_skeleton(), links_stat(), nc(), rand_net(), rand_net_par(), smallworldness()

Examples

```
igraph::make_graph("Walther") %>% net_par()
c_net_index(co_net) -> co_net_with_par
```

`olympic_rings_net` *Plot olympic rings using network*

Description

Plot olympic rings using network

Usage

```
olympic_rings_net()
```

Value

network plot

See Also

Other plot: as.ggig(), c_net_plot(), input_cytoscape(), input_gephi(), netD3plot(), plot.ggig(), twocol_edgelist(), venn_net()

Examples

```
olympic_rings_net()
```

`p.adjust.table` *p.adjust apply on a correlation table (matrix or data.frame)*

Description

p.adjust apply on a correlation table (matrix or data.frame)

Usage

```
p.adjust.table(pp, method = "BH", mode = "all")
```

Arguments

pp	table of p-values
method	see p.adjust, default: "BH".
mode	"all" for all values; "rows" adjust each row one by one; "columns" adjust each column one by one. Default: "all".

Value

a table of adjusted p-values

See Also

Other calculate: c_net_calculate(), cal_sim(), fast_cor(), read_corr()

Examples

```
matrix(abs(rnorm(100, 0.01, 0.1)), 10, 10) -> pp
p.adjust.table(pp, method = "BH", mode = "all") -> pp_adj
```

---

plot.ggig *Plot a ggig*

Description

Plot a ggig

Usage

```
## S3 method for class 'ggig'
plot(
  x,
  coors = NULL,
  ...,
  labels_num = NULL,
  vertex_size_range = NULL,
  edge_width_range = NULL,
  plot_module = FALSE,
  mark_module = FALSE,
  mark_color = NULL,
  mark_alpha = 0.3,
  module_label = FALSE,
  module_label_cex = 2,
  module_label_color = "black",
  module_label_just = c(0.5, 0.5),
  legend_number = FALSE,
  legend = TRUE,
  legend_cex = 1,
```

```
  legend_position = c(left_leg_x = -2, left_leg_y = 1, right_leg_x = 1.2, right_leg_y =
    1),
  group_legend_title = NULL,
  group_legend_order = NULL,
  color_legend = TRUE,
  color_legend_order = NULL,
  size_legend = FALSE,
  size_legend_title = "Node Size",
  edge_legend = TRUE,
  edge_legend_title = "Edge type",
  edge_legend_order = NULL,
  width_legend = FALSE,
  width_legend_title = "Edge width",
  lty_legend = FALSE,
  lty_legend_title = "Edge class",
  lty_legend_order = NULL,
  params_list = NULL,
  seed = 1234
)
```

Arguments

x	ggig object
coors	the coordinates you saved
...	additional parameters for igraph.plotting
labels_num	show how many labels, >1 indicates number, <1 indicates fraction, "all" indicates all.
vertex_size_range	the vertex size range, e.g. c(1,10)
edge_width_range	the edge width range, e.g. c(1,10)
plot_module	logical, plot module?
mark_module	logical, mark the modules?
mark_color	mark color
mark_alpha	mark fill alpha, default 0.3
module_label	show module label?
module_label_cex	module label cex
module_label_color	module label color
module_label_just	module label just, default c(0.5,0.5)
legend_number	legend with numbers
legend	all legends
legend_cex	character expansion factor relative to current par("cex"), default: 1

`legend_position`
: legend_position, default: c(left_leg_x=-1.9,left_leg_y=1,right_leg_x=1.2,right_leg_y=1)

`group_legend_title`
: group_legend_title, length must same to the numbers of v_group

`group_legend_order`
: group_legend_order vector

`color_legend`
: logical

`color_legend_order`
: color_legend_order vector

`size_legend`
: logical

`size_legend_title`
: size_legend_title

`edge_legend`
: logical

`edge_legend_title`
: edge_legend_title

`edge_legend_order`
: edge_legend_order vector, e.g. c("positive","negative")

`width_legend`
: logical

`width_legend_title`
: width_legend_title

`lty_legend`
: logical

`lty_legend_title`
: lty_legend_title

`lty_legend_order`
: lty_legend_order

`params_list`
: a list of parameters, e.g. list(edge_legend = TRUE, lty_legend = FALSE), when the parameter is duplicated, the format argument will be used rather than the argument in params_list.

`seed`
: random seed, default:1234, make sure each plot is the same.

Value

ggplot

See Also

Other plot: `as.ggig()`, `c_net_plot()`, `input_cytoscape()`, `input_gephi()`, `netD3plot()`, `olympic_rings_net()`, `twocol_edgelist()`, `venn_net()`

plot.metanet *Plot a metanet*

Description

Plot a metanet

Usage

```
## S3 method for class 'metanet'
plot(x, ...)
```

Arguments

x	metanet object
...	add

Value

plot

plot.metanet_compare *Plot a metanet_compare*

Description

Plot a metanet_compare

Usage

```
## S3 method for class 'metanet_compare'
plot(x, coors_com = NULL, mains = NULL, ...)
```

Arguments

x	metanet_compare object
coors_com	coors object
mains	a vector of two strings for the main titles of the two networks
...	add

Value

plot

plot.rmt_res *Plot a rmt_res*

Description

Plot a rmt_res

Usage

```
## S3 method for class 'rmt_res'
plot(x, ...)
```

Arguments

x	rmt_res
...	Additional arguments

Value

ggplot

plot.robust *Plot robust*

Description

Plot robust

Usage

```
## S3 method for class 'robust'
plot(
  x,
  indexes = c("Natural_connectivity", "Average_degree"),
  use_ratio = FALSE,
  mode = 1,
  ...
)
```

Arguments

x	robust_test() result (robust object)
indexes	indexes selected to show
use_ratio	use the delete nodes ratio rather than nodes number
mode	plot mode, 1~3
...	additional arguments for group_box

Value

a ggplot

`plot.robustness` *Plot robustness*

Description

Plot robustness

Usage

```
## S3 method for class 'robustness'
plot(x, indexes = "Node_number", ...)
```

Arguments

`x`	`robustness()` result (robustness object)
`indexes`	indexes selected to show
`...`	additional arguments for `group_box`

Value

a ggplot

`plot.vulnerability` *Plot vulnerability*

Description

Plot vulnerability

Usage

```
## S3 method for class 'vulnerability'
plot(x, ...)
```

Arguments

`x`	`vulnerability()` result (vulnerability object)
`...`	add

Value

a ggplot

plot_corr_heatmap *Plot a correlation heatmap*

Description

Plot a correlation heatmap

Usage

```
plot_corr_heatmap(
  corr,
  show.p = FALSE,
  cluster_rows = TRUE,
  cluster_cols = TRUE,
  ...
)
```

Arguments

corr	a corr object, must contain 'r' and 'p.value' matrices.
show.p	whether to show p-values as significance stars on the heatmap.
cluster_rows	whether to cluster rows.
cluster_cols	whether to cluster columns.
...	additional arguments passed to pheatmap::pheatmap.

Value

plot of the correlation heatmap.

plot_e_type_bar *Plot e_type bar*

Description

Plot e_type bar

Usage

```
plot_e_type_bar(go, mode = c("left", "right")[1], degree_threshold = 0)
```

Arguments

go	metanet object
mode	"left" or "right"
degree_threshold	degree threshold

Value

ggplot

Examples

```
data("c_net")
plot_e_type_bar(co_net, degree_threshold = 10)
```

`plot_multi_nets` *Batch drawing multiple network diagrams*

Description

Batch drawing multiple network diagrams

Usage

```
plot_multi_nets(graph_ls, nrow = NULL, ncol = NULL, multi_params_list = NULL)
```

Arguments

`graph_ls`	a list containing igraph objects
`nrow`	nrow
`ncol`	ncol
`multi_params_list`	a list of parameters for each network

Value

No value

Examples

```
plot_multi_nets(list(co_net, co_net2),
  multi_params_list = list(
    list(vertex.color = "skyblue"),
    list(vertex.color = "green3")
  )
)
```

plot_net_degree *Plot degree distribution of networks*

Description

Plot degree distribution of networks

Usage

```
plot_net_degree(gols, net_names = NULL)
```

Arguments

gols	list of metanet
net_names	names of networks

Value

ggplot

print.cohesion *Print method for 'cohesion' objects*

Description

Print method for 'cohesion' objects

Usage

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

Arguments

x	'cohesion' object
...	Additional arguments

Value

No value

print.coors *Print method for 'coors' objects*

Description

Print method for 'coors' objects

Usage

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

Arguments

x	'coors' object
...	additional arguments

Value

No value

print.corr *Print method for 'corr' objects*

Description

Print method for 'corr' objects

Usage

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

Arguments

x	'corr' object
...	additional arguments

Value

No value

print.ggig *Print method for 'ggig' objects*

Description

Print method for 'ggig' objects

Usage

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

Arguments

x	'ggig' object
...	Additional arguments

Value

No value

print.metanet *Print method for 'metanet' objects*

Description

Print method for 'metanet' objects

Usage

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

Arguments

x	'metanet' object
...	Additional arguments

Value

No value

print.metanet_compare *Print method for 'metanet_compare' objects*

Description

Print method for 'metanet_compare' objects

Usage

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

Arguments

x	'metanet_compare' object
...	Additional arguments

Value

No value

print.robust *Print method for 'robust' objects*

Description

Print method for 'robust' objects

Usage

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

Arguments

x	'robust' object
...	Additional arguments

Value

No value

print.robustness *Print method for 'robustness' objects*

Description

Print method for 'robustness' objects

Usage

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

Arguments

x	'robustness' object
...	Additional arguments

Value

No value

print.vulnerability *Print method for 'vulnerability' objects*

Description

Print method for 'vulnerability' objects

Usage

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

Arguments

x	'vulnerability' object
...	Additional arguments

Value

No value

rand_net *Degree distribution comparison with random network*

Description

Degree distribution comparison with random network

Usage

```
rand_net(go = go, plot = TRUE)
```

Arguments

go	igraph object
plot	plot or not

Value

ggplot

See Also

Other topological: `extract_sample_net()`, `fit_power()`, `get_group_skeleton()`, `links_stat()`, `nc()`, `net_par()`, `rand_net_par()`, `smallworldness()`

Examples

```
rand_net(co_net)
```

rand_net_par *Net_pars of many random network*

Description

Net_pars of many random network

Compare some indexes between your net with random networks

Usage

```
rand_net_par(go, reps = 99, threads = 1, verbose = TRUE)

compare_rand(
  pars,
  randp,
  index = c("Average_path_length", "Clustering_coefficient")
)
```

Arguments

go	igraph
reps	simulation time
threads	threads
verbose	verbose
pars	your net pars resulted by net_pars()
randp	random networks pars resulted by rand_net_par()
index	compared indexes: "Average_path_length","Clustering_coefficient" or else

Value

ggplot

See Also

Other topological: `extract_sample_net()`, `fit_power()`, `get_group_skeleton()`, `links_stat()`, `nc()`, `net_par()`, `rand_net()`, `smallworldness()`

Examples

```
data("c_net")
rand_net_par(co_net_rmt, reps = 30) -> randp
net_par(co_net_rmt, fast = FALSE) -> pars
compare_rand(pars, randp)
```

---

read_corr *Read a corr object*

Description

Read a corr object

Usage

```
read_corr(filename)
```

Arguments

filename	filename of .corr

Value

a corr object

See Also

Other calculate: `c_net_calculate()`, `cal_sim()`, `fast_cor()`, `p.adjust.table()`

RMT_threshold *Get RMT threshold for a correlation matrix*

Description

Get RMT threshold for a correlation matrix

Get RMT threshold for a correlation matrix roughly

Usage

```
RMT_threshold(
  occor.r,
  out_dir,
  min_threshold = 0.5,
  max_threshold = 0.8,
  step = 0.02,
  plot = FALSE,
  gif = FALSE,
  verbose = FALSE
)

rmt(occor.r, min_threshold = 0.5, max_threshold = 0.85, step = 0.01)
```

Arguments

occor.r	a corr object or a correlation matrix
out_dir	output dir
min_threshold	min_threshold
max_threshold	max_threshold
step	step
plot	plot pngs?
gif	render a .gif file?
verbose	verbose

Value

a r-threshold

recommend threshold

References

J. Zhou, Y. Deng, FALSE. Luo, Z. He, Q. Tu, X. Zhi, (2010) Functional Molecular Ecological Networks, doi:10.1128/mBio.00169-10. https://matstat.org/content_en/RMT/RMThreshold_Intro.pdf

Examples

```
data(otutab, package = "pcutils")
t(otutab) -> totu
c_net_calculate(totu) -> corr
rmt(corr)
# recommend: 0.69
c_net_build(corr, r_threshold = 0.69) -> co_net_rmt
```

---

save_corr *Save a corr object*

Description

Save a corr object

Usage

```
save_corr(corr, filename = "corr")
```

Arguments

corr	a corr object
filename	filename without extension, default: "corr"

Value

a .corr file

---

show_MetaNet_logo *Show MetaNet logo*

Description

Show MetaNet logo

Usage

```
show_MetaNet_logo()
```

Value

picture

smallworldness *Calculate small-world coefficient*

Description

Calculate small-world coefficient

Usage

```
smallworldness(go, reps = 99, threads = 1, verbose = TRUE)
```

Arguments

go	igraph or metanet
reps	simulation time
threads	threads
verbose	verbose

Value

number

See Also

Other topological: `extract_sample_net()`, `fit_power()`, `get_group_skeleton()`, `links_stat()`, `nc()`, `net_par()`, `rand_net()`, `rand_net_par()`

Examples

```
# set reps at least 99 when you run.
smallworldness(co_net, reps = 9)
```

spatstat_layout *Generate spatial layout using* spatstat

Description

Generate spatial layout using spatstat

Usage

```
spatstat_layout(
  go,
  win,
  type = c("random", "regular"),
  mode = c("surface", "boundary"),
  jitter = 0,
  curved = NULL,
  order_by = NULL,
  order_ls = NULL,
  order_circle = FALSE,
  seed = 1234,
  rescale = TRUE
)
```

Arguments

go	igraph or metanet object
win	A spatstat window object (owin), e.g. disc(), owin(poly=...); Or sf object.
type	Type of distribution: "random", "regular"
mode	"surface", "boundary"
jitter	for surface-regular, defalut 0
curved	Optional curved attribute for coors
order_by	order nodes according to a node attribute
order_ls	manual the discrete variable with a vector, or continuous variable with "desc" to decreasing
order_circle	order nodes from the center of a circle
seed	random seed
rescale	rescale the coordinates to (0,1)

Value

A coors object (data.frame with class "coors" and attribute "curved")

Examples

```
if (requireNamespace("spatstat.geom") && requireNamespace("spatstat.random")) {
  poly_x <- c(0, 2, 2, 0)
  poly_y <- c(0, 0, 1, 1)
  win_poly <- spatstat.geom::owin(poly = list(x = poly_x, y = poly_y))
  plot(win_poly)
  coors1 <- spatstat_layout(co_net, win_poly, type = "regular", mode = "surface")
  plot(co_net, coors = coors1)
  coors2 <- spatstat_layout(co_net2, win_poly, type = "random", mode = "boundary")
  plot(co_net2, coors = coors2)
```

```
  if (requireNamespace("sf")) {
    nc <- sf::st_read(system.file("shape/nc.shp", package = "sf"), quiet = TRUE)
    poly <- nc[1, ]
    coors <- spatstat_layout(go = multi1, win = poly, type = "regular", mode = "surface")
    plot(multi1, coors = coors)
  }
}
```

---

summary.corr *Summary method for 'corr' objects*

Description

Summary method for 'corr' objects

Usage

```
## S3 method for class 'corr'
summary(object, ...)
```

Arguments

object	'corr' object
...	Additional arguments

Value

No value

---

summary_module *Summary module index*

Description

Summary module index

Usage

```
summary_module(go_m, var = "v_class", module = "module", ...)
```

Arguments

go_m	module metanet
var	variable name
module	which column name is module. default: "module"
...	add

Value

ggplot

See Also

Other module: filter_n_module(), get_community(), get_module(), get_module_eigen(), module_detect(), module_eigen(), module_net(), to_module_net(), zp_analyse()

Examples

```
data("c_net")
module_detect(co_net, n_node_in_module = 30) -> co_net_modu
summary_module(co_net_modu, var = "v_class", module = "module")
summary_module(co_net_modu, var = "Abundance", module = "module")
```

---

summ_2col *Summaries two columns information*

Description

Summaries two columns information

Usage

```
summ_2col(df, from = 1, to = 2, count = 3, direct = FALSE)
```

Arguments

df	data.frame
from	first column name or index
to	second column name or index
count	(optional) weight column, if no, each equal to 1
direct	consider direct? default: FALSE

Value

data.frame

Examples

```
test <- data.frame(
  a = sample(letters[1:4], 10, replace = TRUE),
  b = sample(letters[1:4], 10, replace = TRUE)
)
summ_2col(test, direct = TRUE)
summ_2col(test, direct = FALSE)
if (requireNamespace("circlize")) {
  summ_2col(test, direct = TRUE) %>% pcutils::my_circo()
}
```

to_module_net *Transformation a network to a module network*

Description

Transformation a network to a module network

Usage

```
to_module_net(go, edge_type = c("module", "module_from", "module_to")[1])
```

Arguments

go	metanet
edge_type	"module", "module_from", "module_to"

Value

metanet with modules

See Also

Other module: filter_n_module(), get_community(), get_module(), get_module_eigen(), module_detect(), module_eigen(), module_net(), summary_module(), zp_analyse()

transc *MetaNet networks abundance*

Description

MetaNet co_nets

transc_g *MetaNet networks metadata*

Description

MetaNet co_nets

`transform_coors` *Transform the layout of a 'coors' object*

Description

This function applies various transformations to a 'coors' object, including scaling, aspect ratio adjustment, rotation, mirroring, and pseudo-3D perspective transformation.

Usage

```
transform_coors(
  coors,
  scale = 1,
  aspect_ratio = 1,
  rotation = 0,
  mirror_x = FALSE,
  mirror_y = FALSE,
  shear_x = 0,
  shear_y = 0
)
```

Arguments

`coors`	An object of class 'coors', containing node coordinates.
`scale`	A numeric value to scale the layout (default = 1).
`aspect_ratio`	A numeric value to adjust the Y-axis scaling (default = 1).
`rotation`	A numeric value in degrees to rotate the layout (default = 0).
`mirror_x`	A logical value indicating whether to mirror along the X-axis (default = FALSE).
`mirror_y`	A logical value indicating whether to mirror along the Y-axis (default = FALSE).
`shear_x`	A numeric value to apply a shear transformation in the X direction (default = 0).
`shear_y`	A numeric value to apply a shear transformation in the Y direction (default = 0).

Value

A transformed 'coors' object with updated coordinates.

twocol_edgelist *Quick build a metanet from two columns table*

Description

Quick build a metanet from two columns table

Usage

```
twocol_edgelist(edgelist)
```

Arguments

edgelist	two columns table (no elements exist in two columns at same time)

Value

metanet

See Also

Other plot: as.ggig(), c_net_plot(), input_cytoscape(), input_gephi(), netD3plot(), olympic_rings_net(), plot.ggig(), venn_net()

Examples

```
twocol <- data.frame(
  "col1" = sample(letters, 30, replace = TRUE),
  "col2" = sample(c("A", "B"), 30, replace = TRUE)
)
twocol_net <- twocol_edgelist(twocol)
plot(twocol_net)
c_net_plot(twocol_net, g_layout_polygon(twocol_net))
```

venn_net *Venn network*

Description

Venn network

Usage

```
venn_net(tab)
```

Arguments

tab	data.frame (row is elements, column is group), or a list (names is group, value is elements)

Value

plot

See Also

Other plot: as.ggig(), c_net_plot(), input_cytoscape(), input_gephi(), netD3plot(), olympic_rings_net(), plot.ggig(), twocol_edgelist()

Examples

```
data(otutab, package = "pcutils")
tab <- otutab[400:485, 1:3]
venn_net(tab) -> v_net
plot(v_net)
```

---

zp_analyse *Zi-Pi calculate*

Description

Zi-Pi calculate

Zi-Pi plot of vertexes

Usage

```
zp_analyse(go_m, mode = 2, use_origin = TRUE)

zp_plot(go, label = TRUE, mode = 1)
```

Arguments

go_m	igraph object after module_detect()
mode	plot style, 1~3
use_origin	use original_module, default:TRUE, if FALSE, use module
go	igraph object after zp_analyse()
label	show label or not

Value

igraph

a ggplot object

References

1. Guimerà, R. & Amaral, L. Functional cartography of complex metabolic networks. (2005) doi:10.1038/nature03288.

See Also

Other module: filter_n_module(), get_community(), get_module(), get_module_eigen(), module_detect(), module_eigen(), module_net(), summary_module(), to_module_net()

Examples

```
data("c_net")
module_detect(co_net) -> co_net_modu
zp_analyse(co_net_modu) -> co_net_modu
zp_plot(co_net_modu)
zp_plot(co_net_modu, mode = 3)
```

Index