

# Package ‘MetaNet’

January 20, 2025

**Type** Package

**Title** Network Analysis for Omics Data

**Version** 0.1.2

**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** 7.2.3

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

**LazyData** true

**Imports** graphics, dplyr, ggplot2 (>= 3.2.0), ggnewscale, ggrepel, RColorBrewer, grDevices, magrittr, reshape2, stats, tibble, utils, pcutils (>= 0.2.5), rlang

**Suggests** pheatmap, vegan, stringr, foreach, doSNOW, snow, knitr, rmarkdown, prettydoc, Hmisc, gifski, ggraph, networkD3, ggpmisc, ggtree, treeio, circlize, ggpubr

**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] (<<https://orcid.org/0000-0002-9449-7606>>)

**Maintainer** Chen Peng <pengchen2001@zju.edu.cn>

**Repository** CRAN

**Date/Publication** 2024-03-25 20:40:07 UTC

## Contents

anno_edge . . . . .	4
anno_vertex . . . . .	4
arc_count . . . . .	5
arc_taxonomy . . . . .	5
as.ggig . . . . .	6
as_arc . . . . .	6
as_circle_tree . . . . .	7
as_line . . . . .	7
as_polyarc . . . . .	8
as_polycircle . . . . .	9
as_polygon . . . . .	9
cal_sim . . . . .	10
check_tabs . . . . .	11
clean_igraph . . . . .	11
Cohesion . . . . .	12
co_net . . . . .	13
co_net2 . . . . .	13
co_net_rmt . . . . .	13
c_net_annotate . . . . .	14
c_net_build . . . . .	15
c_net_calculate . . . . .	16
c_net_filter . . . . .	17
c_net_from_edgelist . . . . .	18
c_net_layout . . . . .	19
c_net_plot . . . . .	20
c_net_save . . . . .	22
c_net_set . . . . .	23
c_net_stability . . . . .	24
c_net_union . . . . .	26
c_net_update . . . . .	27
df2net_tree . . . . .	27
extract_sub_net . . . . .	28
fast_cor . . . . .	29
filter_n_module . . . . .	29
fit_power . . . . .	31
get_community . . . . .	31
get_e . . . . .	32
get_group_skeleton . . . . .	32
get_module . . . . .	33
get_module_eigen . . . . .	34
get_n . . . . .	34
get_v . . . . .	35
g_layout . . . . .	35
g_layout_polygon . . . . .	37
input_corr . . . . .	38
input_gephi . . . . .	39

is_metanet . . . . .	39
links_stat . . . . .	40
metab . . . . .	41
metab_g . . . . .	41
micro . . . . .	41
micro_g . . . . .	42
module_detect . . . . .	42
module_eigen . . . . .	43
module_net . . . . .	44
multi1 . . . . .	45
multi_net_build . . . . .	45
nc . . . . .	46
netD3plot . . . . .	47
net_par . . . . .	48
olympic_rings_net . . . . .	49
p.adjust.table . . . . .	49
plot.ggig . . . . .	50
plot.metanet . . . . .	52
plot.rmt_res . . . . .	53
plot.robust . . . . .	53
plot.robustness . . . . .	54
plot.vulnerability . . . . .	54
print.cohesion . . . . .	55
print.coors . . . . .	55
print.corr . . . . .	56
print.ggig . . . . .	56
print.metanet . . . . .	57
print.robust . . . . .	57
print.robustness . . . . .	58
print.vulnerability . . . . .	58
rand_net . . . . .	59
rand_net_par . . . . .	59
RMT_threshold . . . . .	60
show_MetaNet_logo . . . . .	61
smallworldness . . . . .	62
summary_module . . . . .	62
summ_2col . . . . .	63
to_module_net . . . . .	64
transc . . . . .	64
transc_g . . . . .	65
twocol_edgelist . . . . .	65
venn_net . . . . .	66
zp_analyse . . . . .	66

---

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\\_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\\_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	<i>Edgelist</i>
-----------	-----------------

---

**Description**

Edgelist for `c_net_from_edgelist()`

---

arc_taxonomy	<i>Edgelist</i>
--------------	-----------------

---

**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\\_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\\_polyarc\(\)](#), [as\\_polycircle\(\)](#), [as\\_polygon\(\)](#), [c\\_net\\_layout\(\)](#)

**Examples**

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

---

as_circle_tree	<i>Layout as a circle_tree</i>
----------------	--------------------------------

---

**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\\_polyarc\(\)](#), [as\\_polycircle\(\)](#), [as\\_polygon\(\)](#), [c\\_net\\_layout\(\)](#)

---

as_line	<i>Layout as a line</i>
---------	-------------------------

---

**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\\_polyarc\(\)](#), [as\\_polycircle\(\)](#), [as\\_polygon\(\)](#), [c\\_net\\_layout\(\)](#)

**Examples**

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

---

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\\_polycircle\(\)](#), [as\\_polygon\(\)](#), [c\\_net\\_layout\(\)](#)

**Examples**

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



---

as_polycircle	<i>Layout as a polycircle</i>
---------------	-------------------------------

---

**Description**

Layout as a polycircle

**Usage**

```
as_polycircle(n = 2)
```

**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\\_polyarc\(\)](#), [as\\_polygon\(\)](#), [c\\_net\\_layout\(\)](#)

**Examples**

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

---

as_polygon	<i>Layout as a polygon</i>
------------	----------------------------

---

**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\\_polyarc\(\)](#), [as\\_polycircle\(\)](#), [c\\_net\\_layout\(\)](#)

**Examples**

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

---

cal\_sim

*Calculate similarity for one t(otutab)*

---

**Description**

Calculate similarity for one t(otutab)

**Usage**

```
cal_sim(totu, method = "bray", norm = FALSE)
```

**Arguments**

totu            t(otutab), row are samples, column are features.  
 method        Dissimilarity index, see [vegdist](#).  
 norm           hellinger normalization in features (default: FALSE).

**Value**

similarity = 1-distance

**See Also**

[vegdist](#)

Other calculate: [c\\_net\\_calculate\(\)](#), [fast\\_cor\(\)](#), [input\\_corr\(\)](#), [p.adjust.table\(\)](#)

**Examples**

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

---

check_tabs	<i>Check tables and extract common samples</i>
------------	--

---

**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	<i>Clean a igraph object</i>
--------------	------------------------------

---

**Description**

Clean a igraph object

**Usage**

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

**Arguments**

go            igraph, metanet objects  
direct        direct?

**Value**

a igraph object

---

Cohesion	<i>Cohesion calculation</i>
----------	-----------------------------

---

### 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 <a href="#">group_box</a> (mode=1) or <a href="#">group_box</a> (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	<i>MetaNet networks</i>
--------	-------------------------

---

**Description**

MetaNet co\_nets

---

co_net2	<i>MetaNet networks</i>
---------	-------------------------

---

**Description**

MetaNet co\_nets

---

co_net_rmt	<i>MetaNet networks</i>
------------	-------------------------

---

**Description**

MetaNet co\_nets

---

c_net_annotate	<i>Annotate a metanet</i>
----------------	---------------------------

---

## 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\\_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	<i>Construct a metanet from a corr object</i>
-------------	---

---

### 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 <code>c_net_calculate()</code> or <code>input_corr()</code> .
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	<i>Calculate correlation for one or two t(otutab), or distance for one t(otutab).</i>
-----------------	---

---

### 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 <a href="#">vegdist</a> .
filename	the prefix of saved .corr file or FALSE.
p.adjust.method	see <a href="#">p.adjust</a>
p.adjust.mode	see <a href="#">p.adjust.table</a>
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\(\)](#), [input\\_corr\(\)](#), [p.adjust.table\(\)](#)



**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_filter	<i>Filter a network according to some attributes</i>
--------------	--

---

**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\\_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 = NULL,  
  direct = FALSE,  
  e_type = NULL,  
  e_class = NULL  
)
```

### Arguments

edgelist	first is source, second is target, others are annotation
vertex	vertex metadata
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 = arc_taxonomy)  
edge_net <- c_net_set(edge_net, vertex_class = "Phylum", edge_width = "n")  
c_net_plot(edge_net)
```

---

c_net_layout	<i>Layout coordinates</i>
--------------	---------------------------

---

### Description

Layout coordinates

### Usage

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

### Arguments

go	igraph or metanet
method	(1) <a href="#">as_line()</a> , <a href="#">as_arc()</a> , <a href="#">as_polygon()</a> , <a href="#">as_polyarc()</a> , <a href="#">as_polycircle()</a> , <a href="#">as_circle_tree()</a> ; (2) <a href="#">as_star()</a> , <a href="#">as_tree()</a> , <a href="#">in_circle()</a> , <a href="#">nicely()</a> , <a href="#">on_grid()</a> , <a href="#">on_sphere()</a> , <a href="#">randomly()</a> , <a href="#">with_dh()</a> , <a href="#">with_fr()</a> , <a href="#">with_gem()</a> , <a href="#">with_graphopt()</a> , <a href="#">with_kk()</a> , <a href="#">with_lgl()</a> , <a href="#">with_mds()</a> , see <a href="#">layout_</a> ; (3) a character, "auto", "backbone", "centrality", "circlepack", "dendrogram", "eigen", "focus", "hive", "igraph", "linear", "manual", "matrix", "partition", "pmds", "stress", "treemap", "unro see <a href="#">create_layout</a>
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 <a href="#">as_line()</a> , <a href="#">as_polygon()</a> .default:0
...	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\\_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\_plot

*Plot a metanet*

---

## Description

Plot a metanet

## Usage

```
c_net_plot(
  go,
  coors = NULL,
  ...,
  labels_num = 5,
  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 = 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,
seed = 1234
)

```

### Arguments

go	an igraph or metanet object
coors	the coordinates you saved
...	additional parameters for <a href="#">igraph.plotting</a>
labels_num	show how many labels,>1 indicates number, <1 indicates fraction, "all" indicates all, default:5
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 colors
mark_alpha	mark fill alpha, default 0.3
module_label	module_label
module_label_cex	module_label_cex
module_label_color	module_label_color
module_label_just	module_label_just
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

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

```

**Value**

a network plot

**See Also**

Other plot: [as.ggig\(\)](#), [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\\_union\(\)](#), [get\\_e\(\)](#), [get\\_n\(\)](#), [get\\_v\(\)](#), [is\\_metanet\(\)](#)

---

c_net_set	<i>Set basic attributes from totu table</i>
-----------	---

---

**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
)
```

**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

**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**

$$V_i = \frac{E - E_i}{E}$$

E is the global efficiency and  $E_i$  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

## Value

metanet

## See Also

Other manipulate: [anno\\_edge\(\)](#), [anno\\_vertex\(\)](#), [c\\_net\\_annotate\(\)](#), [c\\_net\\_filter\(\)](#), [c\\_net\\_save\(\)](#), [get\\_e\(\)](#), [get\\_n\(\)](#), [get\\_v\(\)](#), [is\\_metanet\(\)](#)

## Examples

```
data("c_net")
co_net_union <- c_net_union(co_net, co_net2)
c_net_plot(co_net_union)
```

---

c_net_update	<i>Update a metanet object or transform igraph object to metanet object</i>
--------------	---

---

**Description**

Update a metanet object or transform igraph object to metanet object

**Usage**

```
c_net_update(go, node_break = 5, edge_break = 5)
```

**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

**Value**

metanet

**See Also**

Other build: [c\\_net\\_build\(\)](#), [c\\_net\\_from\\_edgelist\(\)](#), [c\\_net\\_set\(\)](#), [multi\\_net\\_build\(\)](#)

---

df2net_tree	<i>Transform a dataframe to a network edgelist.</i>
-------------	---

---

**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_sub_net	<i>Extract sub-network from the whole network</i>
-----------------	---

---

**Description**

Extract sub-network from the whole network

**Usage**

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

**Arguments**

whole_net	the whole network
otutab	otutab, 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 ?
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\\_par\(\)](#), [rand\\_net\(\)](#), [smallworldness\(\)](#)

**Examples**

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

---

fast_cor	<i>Fast correlation calculation</i>
----------	-------------------------------------

---

**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\(\)](#), [input\\_corr\(\)](#), [p.adjust.table\(\)](#)

**Examples**

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

---

filter_n_module	<i>Filter some modules as others</i>
-----------------	--------------------------------------

---

**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\\_eigen\(\)](#), [get\\_module\(\)](#), [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	<i>Fit power-law distribution for an igraph</i>
-----------	---

---

**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\\_sub\\_net\(\)](#), [get\\_group\\_skeleton\(\)](#), [links\\_stat\(\)](#), [nc\(\)](#), [net\\_par\(\)](#), [rand\\_net\\_par\(\)](#), [rand\\_net\(\)](#), [smallworldness\(\)](#)

**Examples**

```
fit_power(co_net)
```

---

get_community	<i>Get community</i>
---------------	----------------------

---

**Description**

Get community

**Usage**

```
get_community(go_m)
```

**Arguments**

go_m	module metanet
------	----------------

**Value**

community

**See Also**

Other module: [filter\\_n\\_module\(\)](#), [get\\_module\\_eigen\(\)](#), [get\\_module\(\)](#), [module\\_detect\(\)](#), [module\\_eigen\(\)](#), [module\\_net\(\)](#), [summary\\_module\(\)](#), [to\\_module\\_net\(\)](#), [zp\\_analyse\(\)](#)

---

get_e	<i>Get edge information</i>
-------	-----------------------------

---

**Description**

Get edge information

**Usage**

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

**Arguments**

go	metanet object
name	attribute name, default: NULL

**Value**

data.frame

**See Also**

Other manipulate: [anno\\_edge\(\)](#), [anno\\_vertex\(\)](#), [c\\_net\\_annotate\(\)](#), [c\\_net\\_filter\(\)](#), [c\\_net\\_save\(\)](#), [c\\_net\\_union\(\)](#), [get\\_n\(\)](#), [get\\_v\(\)](#), [is\\_metanet\(\)](#)

---

get_group_skeleton	<i>Get skeleton network according to a group</i>
--------------------	--

---

**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, ...)
```



**Arguments**

go	network
Group	vertex column name
count	take which column count, default: NULL
top_N	top_N
ske_net	skeleton
...	additional parameters for <a href="#">igraph.plotting</a>

**Value**

skeleton network

**See Also**

Other topological: [extract\\_sub\\_net\(\)](#), [fit\\_power\(\)](#), [links\\_stat\(\)](#), [nc\(\)](#), [net\\_par\(\)](#), [rand\\_net\\_par\(\)](#), [rand\\_net\(\)](#), [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, name = NULL, simple = FALSE)
```

**Arguments**

go                    metanet object  
name                  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\\_save\(\)](#), [c\\_net\\_union\(\)](#), [get\\_e\(\)](#), [get\\_v\(\)](#), [is\\_metanet\(\)](#)

---

`get_v`*Get vertex information*

---

**Description**

Get vertex information

**Usage**

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

**Arguments**

<code>go</code>	metanet object
<code>name</code>	attribute name, default: NULL

**Value**

data.frame

**See Also**

Other manipulate: [anno\\_edge\(\)](#), [anno\\_vertex\(\)](#), [c\\_net\\_annotate\(\)](#), [c\\_net\\_filter\(\)](#), [c\\_net\\_save\(\)](#), [c\\_net\\_union\(\)](#), [get\\_e\(\)](#), [get\\_n\(\)](#), [is\\_metanet\(\)](#)

---

`g_layout`*Layout with group*

---

**Description**

Layout with group

Layout with group nicely

**Usage**

```

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

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

```

**Arguments**

go	igraph or metanet
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 <a href="#">c_net_layout</a> default: <a href="#">in_circle()</a>
zoom1	big network layout size
layout2	one of functions: layout method for <a href="#">c_net_layout</a> , 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.
...	add
mode	circlepack, treemap, backbone, stress

**Value**

coors

**See Also**

Other g\_layout: [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)

```

```
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

### 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
)
```

**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

**Value**

coors

**See Also**

Other g\_layout: [g\\_layout\(\)](#)

**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)
```

---

input\_corr

*Import corr from .csv file*

---

**Description**

Import corr from .csv file

**Usage**

```
input_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\(\)](#)

---

input_gephi	<i>Input a graphml file exported by Gephi</i>
-------------	---

---

**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\(\)](#), [netD3plot\(\)](#), [olympic\\_rings\\_net\(\)](#), [plot.ggig\(\)](#), [twocol\\_edgelist\(\)](#), [venn\\_net\(\)](#)

---

is_metanet	<i>Is this object a metanet object?</i>
------------	---

---

**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\\_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 = 6,
  colors = NULL,
  legend_number = FALSE,
  legend = TRUE,
  legend_cex = 1,
  legend_position = c(left_leg_x = -1.6, left_leg_y = 1, right_leg_x = 1.2, right_leg_y =
    1),
  col_legend_order = NULL,
  group_legend_title = NULL,
  group_legend_order = NULL
)
```

**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:5
colors	colors
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)  
 col\_legend\_order  
     col\_legend\_order vector,  
 group\_legend\_title  
     group\_legend\_title, length must same to the numbers of v\_group  
 group\_legend\_order  
     group\_legend\_order vector

**Value**

plot

**See Also**

Other topological: [extract\\_sub\\_net\(\)](#), [fit\\_power\(\)](#), [get\\_group\\_skeleton\(\)](#), [nc\(\)](#), [net\\_par\(\)](#), [rand\\_net\\_par\(\)](#), [rand\\_net\(\)](#), [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")
}

```

---

metab	<i>MetaNet networks abundance</i>
-------	-----------------------------------

---

**Description**

MetaNet co\_nets

---

metab_g	<i>MetaNet networks metadata</i>
---------	----------------------------------

---

**Description**

MetaNet co\_nets

---

micro	<i>MetaNet networks abundance</i>
-------	-----------------------------------

---

**Description**

MetaNet co\_nets

---

micro_g	<i>MetaNet networks metadata</i>
---------	----------------------------------

---

**Description**

MetaNet co\_nets

---

module_detect	<i>Detect the modules</i>
---------------	---------------------------

---

**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\\_eigen\(\)](#), [get\\_module\(\)](#), [module\\_eigen\(\)](#), [module\\_net\(\)](#), [summary\\_module\(\)](#), [to\\_module\\_net\(\)](#), [zp\\_analyse\(\)](#)

**Examples**

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

---

module_eigen	<i>Calculate the eigenvalue of each module and correlation of nodes and eigenvalue (node_eigen_cor).</i>
--------------	--

---

### 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 <a href="#">facet_wrap</a> , 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\\_eigen\(\)](#), [get\\_module\(\)](#), [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	<i>Generate a n-modules network</i>
------------	-------------------------------------

---

**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\\_eigen\(\)](#), [get\\_module\(\)](#), [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 = 20))
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 <a href="#">p.adjust</a>

**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\\_sub\\_net\(\)](#), [fit\\_power\(\)](#), [get\\_group\\_skeleton\(\)](#), [links\\_stat\(\)](#), [net\\_par\(\)](#), [rand\\_net\\_par\(\)](#), [rand\\_net\(\)](#), [smallworldness\(\)](#)

**Examples**

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

---

netD3plot	<i>plot use networkD3</i>
-----------	---------------------------

---

**Description**

plot use networkD3

**Usage**

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

**Arguments**

go	metanet
v_class	which attributes use to be v_class
...	see <a href="#">forceNetwork</a>

**Value**

D3 plot

**See Also**

Other plot: [as.ggig\(\)](#), [c\\_net\\_plot\(\)](#), [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)
```

```
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 ?
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\\_sub\\_net\(\)](#), [fit\\_power\(\)](#), [get\\_group\\_skeleton\(\)](#), [links\\_stat\(\)](#), [nc\(\)](#), [rand\\_net\\_par\(\)](#), [rand\\_net\(\)](#), [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\\_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 <a href="#">p.adjust</a> , 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\(\)](#), [input\\_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 = 5,
  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,
seed = 1234
)

```

### Arguments

x	ggig object
coors	the coordinates you saved
...	additional parameters for <a href="#">igraph.plotting</a>
labels_num	show how many labels,>1 indicates number, <1 indicates fraction, "all" indicates all, default:5
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 colors
mark_alpha	mark fill alpha, default 0.3
module_label	module_label
module_label_cex	module_label_cex
module_label_color	module_label_color
module_label_just	module_label_just
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
seed                   random seed, default:1234, make sure each plot is the same.

```

**Value**

ggplot

**See Also**

Other plot: [as.ggig\(\)](#), [c\\_net\\_plot\(\)](#), [input\\_gephi\(\)](#), [netD3plot\(\)](#), [olympic\\_rings\\_net\(\)](#), [twocol\\_edgelist\(\)](#), [venn\\_net\(\)](#)

---

plot.metanet	<i>Plot a metanet</i>
--------------	-----------------------

---

**Description**

Plot a metanet

**Usage**

```

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

```

**Arguments**

```

x          metanet object
...       add

```

**Value**

plot

---

plot.rmt_res	<i>Plot a rmt_res</i>
--------------	-----------------------

---

**Description**

Plot a rmt\_res

**Usage**

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

**Arguments**

x	rmt_res
...	Additional arguments

**Value**

ggplot

---

plot.robust	<i>Plot robust</i>
-------------	--------------------

---

**Description**

Plot robust

**Usage**

```
## S3 method for class 'robust'
plot(
  x,
  indexes = c("Natural_connectivity", "Average_path_length", "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 <a href="#">group_box</a>

**Value**

a ggplot

---

plot.robustness	<i>Plot robustness</i>
-----------------	------------------------

---

**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 <a href="#">group_box</a>

**Value**

a ggplot

---

plot.vulnerability	<i>Plot vulnerability</i>
--------------------	---------------------------

---

**Description**

Plot vulnerability

**Usage**

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

**Arguments**

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

**Value**

a ggplot

---

print.cohesion	<i>Print method for 'cohesion' objects</i>
----------------	--

---

**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	<i>Print method for 'coors' objects</i>
-------------	---

---

**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	<i>Print method for 'metanet' objects</i>
---------------	---

---

**Description**

Print method for 'metanet' objects

**Usage**

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

**Arguments**

x	'metanet' object
...	Additional arguments

**Value**

No value

---

print.robust	<i>Print method for 'robust' objects</i>
--------------	--

---

**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)
```

**Arguments**

`go` igraph object

**Value**

ggplot

**See Also**

Other topological: [extract\\_sub\\_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\\_sub\\_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)
```

---

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(
  occur.r,
  out_dir,
  min_threshold = 0.5,
  max_threshold = 0.8,
  step = 0.02,
  gif = FALSE,
  verbose = FALSE
)

rmt(occur.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
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](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
```

---

show_MetaNet_logo	<i>Show MetaNet logo</i>
-------------------	--------------------------

---

**Description**

Show MetaNet logo

**Usage**

```
show_MetaNet_logo()
```

**Value**

picture

---

smallworldness	<i>Calculate small-world coefficient</i>
----------------	--

---

**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\\_sub\\_net\(\)](#), [fit\\_power\(\)](#), [get\\_group\\_skeleton\(\)](#), [links\\_stat\(\)](#), [nc\(\)](#), [net\\_par\(\)](#), [rand\\_net\\_par\(\)](#), [rand\\_net\(\)](#)

**Examples**

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

---

summary_module	<i>Summary module index</i>
----------------	-----------------------------

---

**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\\_eigen\(\)](#), [get\\_module\(\)](#), [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)
```

**Arguments**

go                    metanet

**Value**

metanet with modules

**See Also**

Other module: [filter\\_n\\_module\(\)](#), [get\\_community\(\)](#), [get\\_module\\_eigen\(\)](#), [get\\_module\(\)](#), [module\\_detect\(\)](#), [module\\_eigen\(\)](#), [module\\_net\(\)](#), [summary\\_module\(\)](#), [zp\\_analyse\(\)](#)

---

transc

*MetaNet networks abundance*


---

**Description**

MetaNet co\_nets



---

transc_g	<i>MetaNet networks metadata</i>
----------	----------------------------------

---

**Description**

MetaNet co\_nets

---

twocol_edgelist	<i>Quick build a metanet from two columns table</i>
-----------------	---

---

**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\\_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	<i>Venn network</i>
----------	---------------------

---

**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\\_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	<i>Zi-Pi calculate</i>
------------	------------------------

---

**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\\_eigen\(\)](#), [get\\_module\(\)](#), [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
if (requireNamespace("ggrepel")) {
  zp_plot(co_net_modu)
  zp_plot(co_net_modu, mode = 3)
}
```

# Index

- \* **build**
    - c\_net\_build, 15
    - c\_net\_from\_edgelist, 18
    - c\_net\_set, 23
    - c\_net\_update, 27
    - multi\_net\_build, 45
  - \* **calculate**
    - c\_net\_calculate, 16
    - cal\_sim, 10
    - fast\_cor, 29
    - input\_corr, 38
    - p.adjust.table, 49
  - \* **g\_layout**
    - g\_layout, 35
    - g\_layout\_polygon, 37
  - \* **layout**
    - as\_arc, 6
    - as\_circle\_tree, 7
    - as\_line, 7
    - as\_polyarc, 8
    - as\_polycircle, 9
    - as\_polygon, 9
    - c\_net\_layout, 19
  - \* **manipulate**
    - anno\_edge, 4
    - anno\_vertex, 4
    - c\_net\_annotate, 14
    - c\_net\_filter, 17
    - c\_net\_save, 22
    - c\_net\_union, 26
    - get\_e, 32
    - get\_n, 34
    - get\_v, 35
    - is\_metanet, 39
  - \* **module**
    - filter\_n\_module, 29
    - get\_community, 31
    - get\_module, 33
    - get\_module\_eigen, 34
    - module\_detect, 42
    - module\_eigen, 43
    - module\_net, 44
    - summary\_module, 62
    - to\_module\_net, 64
    - zp\_analyse, 66
  - \* **plot**
    - as.ggig, 6
    - c\_net\_plot, 20
    - input\_gephi, 39
    - netD3plot, 47
    - olympic\_rings\_net, 49
    - plot.ggig, 50
    - twocol\_edgelist, 65
    - venn\_net, 66
  - \* **topological**
    - extract\_sub\_net, 28
    - fit\_power, 31
    - get\_group\_skeleton, 32
    - links\_stat, 40
    - nc, 46
    - net\_par, 48
    - rand\_net, 59
    - rand\_net\_par, 59
    - smallworldness, 62
- anno\_edge, 4, 5, 14, 17, 23, 26, 32, 35, 40  
anno\_node (anno\_vertex), 4  
anno\_vertex, 4, 4, 14, 17, 23, 26, 32, 35, 40  
arc\_count, 5  
arc\_taxonomy, 5  
as.ggig, 6, 22, 39, 47, 49, 52, 65, 66  
as.metanet (c\_net\_update), 27  
as\_arc, 6, 7–10, 19  
as\_circle\_tree, 7, 7, 8–10, 19  
as\_line, 7, 7, 8–10, 19  
as\_polyarc, 7, 8, 8, 9, 10, 19  
as\_polycircle, 7, 8, 9, 10, 19  
as\_polygon, 7–9, 9, 19

- c\_net\_annotate, 4, 5, 14, 17, 23, 26, 32, 35, 40
- c\_net\_build, 15, 18, 24, 27, 46
- c\_net\_cal (c\_net\_calculate), 16
- c\_net\_calculate, 10, 16, 29, 39, 50
- c\_net\_filter, 4, 5, 14, 17, 23, 26, 32, 35, 40
- c\_net\_from\_edgelist, 15, 18, 24, 27, 46
- c\_net\_index (net\_par), 48
- c\_net\_lay (c\_net\_layout), 19
- c\_net\_layout, 7–10, 19, 36
- c\_net\_module (module\_detect), 42
- c\_net\_plot, 6, 20, 39, 47, 49, 52, 65, 66
- c\_net\_save, 4, 5, 14, 17, 22, 26, 32, 35, 40
- c\_net\_set, 15, 18, 23, 27, 46
- c\_net\_stability, 24
- c\_net\_union, 4, 5, 14, 17, 23, 26, 32, 35, 40
- c\_net\_update, 15, 18, 24, 27, 46
- cal\_sim, 10, 16, 29, 39, 50
- check\_tabs, 11
- clean\_igraph, 11
- co\_net, 13
- co\_net2, 13
- co\_net\_rmt, 13
- Cohesion, 12
- combine\_n\_module (filter\_n\_module), 29
- compare\_rand (rand\_net\_par), 59
- create\_layout, 19
  
- df2net\_tree, 27
  
- extract\_sub\_net, 28, 31, 33, 41, 47, 48, 59, 60, 62
  
- facet\_wrap, 43
- fast\_cor, 10, 16, 29, 39, 50
- filter\_n\_module, 29, 32–34, 42–44, 63, 64, 67
- fit\_power, 28, 31, 33, 41, 47, 48, 59, 60, 62
- forceNetwork, 47
  
- g\_layout, 35, 38
- g\_layout\_nice (g\_layout), 35
- g\_layout\_polyarc (g\_layout\_polygon), 37
- g\_layout\_polycircle (g\_layout\_polygon), 37
- g\_layout\_polygon, 36, 37
- get\_community, 30, 31, 33, 34, 42–44, 63, 64, 67
- get\_e, 4, 5, 14, 17, 23, 26, 32, 35, 40
- get\_group\_skeleton, 28, 31, 32, 41, 47, 48, 59, 60, 62
- get\_module, 30, 32, 33, 34, 42–44, 63, 64, 67
- get\_module\_eigen, 30, 32, 33, 34, 42–44, 63, 64, 67
- get\_n, 4, 5, 14, 17, 23, 26, 32, 34, 35, 40
- get\_v, 4, 5, 14, 17, 23, 26, 32, 35, 35, 40
- group\_box, 12, 53, 54
  
- igraph.plotting, 21, 33, 51
- input\_corr, 10, 16, 29, 38, 50
- input\_gephi, 6, 22, 39, 47, 49, 52, 65, 66
- is.metanet (is\_metanet), 39
- is\_metanet, 4, 5, 14, 17, 23, 26, 32, 35, 39
  
- layout\_, 19
- links\_stat, 28, 31, 33, 40, 47, 48, 59, 60, 62
  
- metab, 41
- metab\_g, 41
- micro, 41
- micro\_g, 42
- module\_detect, 30, 32–34, 42, 43, 44, 63, 64, 67
- module\_eigen, 30, 32–34, 42, 43, 44, 63, 64, 67
- module\_expression (module\_eigen), 43
- module\_net, 30, 32–34, 42, 43, 44, 63, 64, 67
- multi1, 45
- multi\_net\_build, 15, 18, 24, 27, 45
  
- nc, 28, 31, 33, 41, 46, 48, 59, 60, 62
- net\_par, 28, 31, 33, 41, 47, 48, 59, 60, 62
- netD3plot, 6, 22, 39, 47, 49, 52, 65, 66
  
- olympic\_rings\_net, 6, 22, 39, 47, 49, 52, 65, 66
  
- p.adjust, 16, 45, 49
- p.adjust.table, 10, 16, 29, 39, 49
- plot.cohesion (Cohesion), 12
- plot.ggig, 6, 22, 39, 47, 49, 50, 65, 66
- plot.metanet, 52
- plot.rmt\_res, 53
- plot.robust, 53
- plot.robustness, 54
- plot.vulnerability, 54
- plot\_module\_tree (filter\_n\_module), 29
- print.cohesion, 55
- print.coors, 55

`print.corr`, 56  
`print.ggig`, 56  
`print.metanet`, 57  
`print.robust`, 57  
`print.robustness`, 58  
`print.vulnerability`, 58

`rand_net`, 28, 31, 33, 41, 47, 48, 59, 60, 62  
`rand_net_par`, 28, 31, 33, 41, 47, 48, 59, 59, 62

`rmt (RMT_threshold)`, 60  
`RMT_threshold`, 60  
`robust_test (c_net_stability)`, 24  
`robustness (c_net_stability)`, 24

`show_MetaNet_logo`, 61  
`skeleton_plot (get_group_skeleton)`, 32  
`smallworldness`, 28, 31, 33, 41, 47, 48, 59, 60, 62

`summ_2col`, 63  
`summary_module`, 30, 32–34, 42–44, 62, 64, 67

`to_module_net`, 30, 32–34, 42–44, 63, 64, 67  
`transc`, 64  
`transc_g`, 65  
`twocol_edgelist`, 6, 22, 39, 47, 49, 52, 65, 66

`vegdist`, 10, 16  
`venn_net`, 6, 22, 39, 47, 49, 52, 65, 66  
`vulnerability (c_net_stability)`, 24

`zp_analyse`, 30, 32–34, 42–44, 63, 64, 66  
`zp_plot (zp_analyse)`, 66