Package ‘MetaNet’
March 25, 2024

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,
RColourBrewer, grDevices, magrittr, reshape2, stats, tibble,
utils, putils (>= 0.2.5), rlang
Suggests pheatmap, vegan, stringr, foreach, doSNOW, snow, knitr,
markdown, prettydoc, Hmisc, gdfs, 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
## R topics documented:

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

- `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
- `multi` ..................................................................... 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

### Index

68
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)
**arc_count**

**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**

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

---

<table>
<thead>
<tr>
<th>arc_count</th>
<th>Edgelist</th>
</tr>
</thead>
</table>

**Description**

Edgelist for `c_net_from_edgelist()`

---

<table>
<thead>
<tr>
<th>arc_taxonomy</th>
<th>Edgelist</th>
</tr>
</thead>
</table>

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

---

**as_circle_tree**  
*Layout as a circle_tree*

Description
Layout as a circle_tree

Usage
```r
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**  
*Layout as a line*

Description
Layout as a line

Usage
```r
as_line(angle = 0)
```
as_polyarc

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_polyarc(co_net)

c_net_plot(co_net, coors = as_polyarc(pi / 2))
as_polycircle

Layout as a polycircle

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

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
cal_sim

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

**Description**

Check tables and extract common samples

**Usage**

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

**Arguments**

- `...`
- `tables`

**Value**

formatted tables

**Examples**

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

---

### clean_igraph

**Description**

Clean a igraph object

**Usage**

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

**Arguments**

- `go`
- `direct`

**Value**

a igraph object
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

### 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_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

```r
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 `input_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

```r
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**

```r
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()`, `input_corr()`, `p.adjust.table()`
**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_save(), c_net_union(), get_e(), get_n(), get_v(), is_metanet()

**Examples**

\[
data("multi\_net")
c\_net\_filter(multi1, v\_group %in% c("omic1", "omic2"))
\]
Construct a network from edge_list dataframe

Usage

```r
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

```r
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**

*Layout coordinates*

**Description**

Layout coordinates

**Usage**

```r
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) `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_`;
  - 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
- **...**
  - 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

```r
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

```r
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,
)```
c_net_plot

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 igraph.plotting
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
Description

Save network file

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(multi)

c_net_save(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_anotate(), c_net_filter(), 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

```r
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)
c_net_stability

Evaluate the stability of a network

Description

\[ V_i = \frac{E - E_i}{E} \]

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,
)
c_net_stability

    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)

<table>
<thead>
<tr>
<th>c_net_union</th>
<th>Union two networks</th>
</tr>
</thead>
</table>

**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**

*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)`

**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**

*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

```r
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  

Extract sub-network from the whole network

Description

Extract sub-network from the whole network

Usage

```r
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

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

**Fast correlation calculation**

**Description**

Fast correlation calculation

**Usage**

```r
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**

```r
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_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

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_sub_net(), get_group_skeleton(), links_stat(), nc(), net_par(), rand_net_par(), rand_net(), smallworldness()

Examples
fit_power(co_net)

gt_community

Get community

Description
Get community

Usage
gt_community(go_m)

Arguments

- go_m: module metanet

Value

community
**get_e**

*Get edge information*

### Description

Get edge information

### Usage

```r
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**

*Get skeleton network according to a group*

### Description

Get skeleton network according to a group

### Usage

```r
group_skeleton(go, Group = "v_class", count = NULL, top_N = 8)
skeleton_plot(ske_net, ...)
```
get_module

Arguments

- **go**
  - network
- **Group**
  - vertex column name
- **count**
  - take which column count, default: NULL
- **top_N**
  - top_N
- **ske_net**
  - skeleton
- ... additional parameters for *igraph.plotting*

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

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

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
get_v

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()

---

**get_v**  
*Get vertex information*

---

**Description**

Get vertex information

**Usage**

```r
get_v(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_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 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.
add
mode circlepack, treemap, backbone, stress

Value
coops

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 polyarc

#### Usage

```r

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
)
```
Description
Import corr from .csv file

Usage
input_corr(filename)

Arguments
filename filename of .corr

Value
a corr object
**input_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 calculate: `c_net_calculate()`, `cal_sim()`, `fast_cor()`, `p.adjust.table()`

---

**is_metanet**

**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

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

---

**links_stat**  
*Link summary of the network*

**Description**

Link summary of the network

**Usage**

```r
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
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

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_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

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_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

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_eigen(), get_module(), module_detect(), module_eigen(), summary_module(), to_module_net(), zp_analyse()
Examples

```r
  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**

---

### multi_net_build

**Multi-omics network build**

Description

MetaNet co_nets

Usage

```r
  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`
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

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

netD3plot

plot use networkD3

Description

plot use networkD3

Usage

```r
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_gephi()`, `olympic_rings_net()`, `plot.ggig()`, `twocol_edgelist()`, `venn_net()`

Examples

```r
data("c.net")
plot(co_net2)
if (requireNamespace("networkD3")) {
  netD3plot(co_net2)
}
```
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: indexes of the whole network
- v_index: indexes of each vertex
- e_index: indexes 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

```r
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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pp</td>
<td>table of p-values</td>
</tr>
<tr>
<td>method</td>
<td>see p.adjust, default: &quot;BH&quot;.</td>
</tr>
<tr>
<td>mode</td>
<td>&quot;all&quot; for all values; &quot;rows&quot; adjust each row one by one; &quot;columns&quot; adjust each column one by one. Default: &quot;all&quot;.</td>
</tr>
</tbody>
</table>

Value
a table of adjusted p-values
plot.ggig

See Also

Other calculate: `c_net_calculate()`, `cal_sim()`, `fast_cor()`, `input_corr()`

Examples

```r
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

```r
## 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(0.5, 0.5),
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 igraph.plotting
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

edge_legend

edge_legend_title

edge_legend_order

width_legend

width_legend_title

lty_legend

lty_legend_title

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()

Description

Plot a metanet

Usage

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

Arguments

x metanet object

... 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_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 group_box
plot.robustness  

**Description**

Plot robustness

**Usage**

```r
## 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  

**Description**

Plot vulnerability

**Usage**

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

**Arguments**

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

**Value**

a ggplot
print.cohesion

### Description
Print method for 'cohesion' objects

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

### Arguments
- `x`: 'cohesion' object
- `...`: Additional arguments

### Value
No value

print.coors

### Description
Print method for 'coors' objects

### Usage
```r
## 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**
```r
## 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**
```r
## 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

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

Arguments

- `x`: 'metanet' object
- `...`: Additional arguments

Value

No value

---

print.robust

Print method for 'robust' objects

Description

Print method for 'robust' objects

Usage

```r
## 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**

```r
## 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**

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

**Arguments**

- `x`  
  'vulnerability' object
- `...`  
  Additional arguments

**Value**

No value
**Description**

Degree distribution comparison with random network

**Usage**

```r
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**

```r
rand_net(co_net)
```

---

**Description**

Net_pars of many random network

Compare some indexes between your net with random networks

**Usage**

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

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

Get RMT threshold for a correlation matrix

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_co_rmt, reps = 30) -> randp
net_par(co_co_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(
    occor.r,
    out_dir,
    min_threshold = 0.5,
    max_threshold = 0.8,
    step = 0.02,
    gif = FALSE,
    verbose = FALSE
)

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

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


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

Description

Show MetaNet logo

Usage

show_MetaNet_logo()

Value

picture
smallworldness  

*Calculate small-world coefficient*

**Description**

Calculate small-world coefficient

**Usage**

```r
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**

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

summary_module  

*Summary module index*

**Description**

Summary module index

**Usage**

```r
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_g

MetaNet networks metadata

Description

MetaNet co_nets

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_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  

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  

Description
Zi-Pi calculate

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


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

```r
data("c\_net")
module\_detect(co\_net) \rightarrow co\_net\_modu
zp\_analyse(co\_net\_modu) \rightarrow 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, 9, 10, 19
as_polycircle, 7, 8, 9, 10, 19
as_polygon, 7–9, 9, 19
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, 60, 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