

# Package ‘meconetcomp’

June 23, 2024

**Type** Package

**Title** Compare Microbial Networks of 'trans\_network' Class of 'microeco' Package

**Version** 0.5.1

**Author** Chi Liu [aut, cre],  
Minjie Yao [ctb],  
Xiangzhen Li [ctb]

**Maintainer** Chi Liu <liuchi0426@126.com>

**Description** Compare microbial co-occurrence networks created from 'trans\_network' class of 'microeco' package <<https://github.com/ChiLiubio/microeco>>. This package is the extension of 'trans\_network' class of 'microeco' package and especially useful when different networks are constructed and analyzed simultaneously.

**URL** <https://github.com/ChiLiubio/meconetcomp>

**Depends** R (>= 3.5.0)

**Imports** R6, microeco (>= 1.3.0), magrittr, dplyr, igraph, reshape2, ggpubr, ggplot2

**Suggests** rgexf, ape, file2meco, agricolae

**License** GPL-3

**LazyData** true

**Encoding** UTF-8

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2024-06-23 02:20:02 UTC

**RoxygenNote** 7.3.1

## Contents

cal_module	2
cal_network_attr	3

cohesionclass . . . . .	4
edge_comp . . . . .	6
edge_node_distance . . . . .	7
edge_tax_comp . . . . .	9
get_edge_table . . . . .	10
get_node_table . . . . .	11
meconetcomp . . . . .	11
node_comp . . . . .	12
robustness . . . . .	13
soil_amp . . . . .	16
soil_amp_network . . . . .	16
soil_measure_diversity . . . . .	16
stool_met . . . . .	17
subnet_property . . . . .	17
subset_network . . . . .	18
vulnerability . . . . .	19
<b>Index</b>	<b>20</b>

---

cal_module	<i>Assign modules to each network</i>
------------	---------------------------------------

---

## Description

Calculating modularity of networks and assign the modules to nodes for each network.

## Usage

```
cal_module(
  network_list,
  undirected_method = "cluster_fast_greedy",
  directed_method = "cluster_optimal",
  ...
)
```

## Arguments

network_list	a list with multiple networks; all the networks should be trans_network object created from trans_network class of microeco package.
undirected_method	default "cluster_fast_greedy"; the modularity algorithm for undirected network; see cal_module function of trans_network class for more algorithms.
directed_method	default 'cluster_optimal'; the modularity algorithm for directed network.
...	other parameters (except for method) passed to cal_module function of trans_network class.

**Value**

list, with module attribute in nodes of each network

**Examples**

```
data(soil_amp_network)
soil_amp_network <- cal_module(soil_amp_network)
```

---

cal_network_attr	<i>Calculate network topological property for each network</i>
------------------	--

---

**Description**

Calculate the topological properties of all the networks and merge the results into one table.

**Usage**

```
cal_network_attr(network_list)
```

**Arguments**

network\_list a list with multiple networks; all the networks should be trans\_network object created from trans\_network class of microeco package.

**Value**

data.frame

**Examples**

```
data(soil_amp_network)
test <- cal_network_attr(soil_amp_network)
```

---

 cohesionclass

 Calculate the cohesion of samples for each network
 

---

## Description

The cohesion is a method for quantifying the connectivity of microbial communities <doi:10.1038/ismej.2017.91>. It is defined:

$$C_j^{pos} = \sum_{i=1}^n a_i \cdot \bar{r}_{i|r>0}$$

$$C_j^{neg} = \sum_{i=1}^n a_i \cdot \bar{r}_{i|r<0}$$

where  $C_j^{pos}$  is the positive cohesion, and  $C_j^{neg}$  is the negative cohesion.  $a_i$  is the relative abundance of species  $i$  in sample  $j$ .  $\bar{r}_{i|r>0}$  denotes the mean weight (correlation coefficient, interaction strength) of all the edges (related with species  $i$ ) with positive association.

## Methods

### Public methods:

- `cohesionclass$new()`
- `cohesionclass$cal_diff()`
- `cohesionclass$plot()`
- `cohesionclass$clone()`

### Method `new()`:

*Usage:*

```
cohesionclass$new(network_list)
```

*Arguments:*

`network_list` a list with multiple networks; all the networks should be `trans_network` object created from `trans_network` class of `microeco` package.

*Returns:* `res_list`, stored in the object. It includes two tables: `res_feature` and `res_sample`. In `res_feature`, the `r_pos` and `r_neg` columns mean the  $\bar{r}_{i|r>0}$  and  $\bar{r}_{i|r<0}$ . In `res_sample`, the `c_pos` and `c_neg` columns denote  $C_j^{pos}$  and  $C_j^{neg}$ .

*Examples:*

```
t1 <- cohesionclass$new(soil_amp_network)
```

### Method `cal_diff()`: Differential test.

*Usage:*

```
cohesionclass$cal_diff(
  measure = "c_pos",
  method = c("anova", "KW", "KW_dunn", "wilcox", "t.test")[1],
  ...
)
```

*Arguments:*

measure default "c\_pos"; "c\_pos" or "c\_neg" in the res\_list\$sample; "r\_pos" or "r\_neg" in the res\_list\$feature.

method default "anova"; see the following available options:

'anova' Duncan's multiple range test for anova

'KW' KW: Kruskal-Wallis Rank Sum Test for all groups ( $\geq 2$ )

'KW\_dunn' Dunn's Kruskal-Wallis Multiple Comparisons, see dunnTest function in FSA package

'wilcox' Wilcoxon Rank Sum and Signed Rank Tests for all paired groups

't.test' Student's t-Test for all paired groups

... parameters passed to cal\_diff function of trans\_alpha class of microeco package.

*Returns:* res\_diff in object. See the Return of cal\_diff function in trans\_alpha class of microeco package.

*Examples:*

```
\donttest{
t1$cal_diff(method = "wilcox")
}
```

**Method** plot(): Plot the result.

*Usage:*

```
cohesionclass$plot(measure = "c_pos", ...)
```

*Arguments:*

measure default "c\_pos"; "c\_pos" or "c\_neg" in the res\_list\$sample; "r\_pos" or "r\_neg" in the res\_list\$feature.

... parameters pass to plot\_alpha function of trans\_alpha class of microeco package.

*Returns:* ggplot.

*Examples:*

```
\donttest{
t1$plot(boxplot_add = "none", add_sig = TRUE)
}
```

**Method** clone(): The objects of this class are cloneable with this method.

*Usage:*

```
cohesionclass$clone(deep = FALSE)
```

*Arguments:*

deep Whether to make a deep clone.

**Examples**

```
## -----
## Method `cohesionclass$new`
## -----

t1 <- cohesionclass$new(soil_amp_network)
```

```
## -----
## Method `cohesionclass$cal_diff`
## -----

t1$cal_diff(method = "wilcox")

## -----
## Method `cohesionclass$plot`
## -----

t1$plot(boxplot_add = "none", add_sig = TRUE)
```

---

edge_comp	<i>Generate a microtable object with paired nodes distributions of edges across networks</i>
-----------	--

---

### Description

Generate a `microtable` object with paired nodes distributions of edges across networks. Useful for the edge comparisons across different networks. The return `otu_table` in `microtable` object has the binary numbers in which 1 represents the presence of the edge in the corresponding network.

### Usage

```
edge_comp(network_list)
```

### Arguments

`network_list` a list with multiple networks; all the networks should be `trans_network` object created from `trans_network` class of `microeco` package.

### Value

`microtable` object

### Examples

```
data(soil_amp_network)
test <- edge_comp(soil_amp_network)
# test is a microtable object
```

---

edge_node_distance	<i>Perform the distance distribution of paired nodes in edges across networks.</i>
--------------------	--

---

## Description

This class is a wrapper for a series of analysis on the distance values of paired nodes in edges across networks, including distance matrix conversion, the differential test and the visualization.

## Methods

### Public methods:

- [edge\\_node\\_distance\\$new\(\)](#)
- [edge\\_node\\_distance\\$cal\\_diff\(\)](#)
- [edge\\_node\\_distance\\$plot\(\)](#)
- [edge\\_node\\_distance\\$clone\(\)](#)

### Method new():

*Usage:*

```
edge_node_distance$new(
  network_list,
  dis_matrix = NULL,
  label = "+",
  with_module = FALSE,
  module_thres = 2
)
```

*Arguments:*

`network_list` a list with multiple networks; all the networks should be `trans_network` object created from `trans_network` class of `microeco` package.

`dis_matrix` default `NULL`; the distance matrix of nodes, used for the value extraction; must be a symmetrical matrix with both `colnames` and `rownames` (i.e. feature names).

`label` default `"+"`; `"+"` or `"-"` or `c("+", "-")`; the edge label used for the selection of edges.

`with_module` default `FALSE`; whether show the module classification of nodes in the result.

`module_thres` default `2`; the threshold of the nodes number of modules remained when `with_module = TRUE`.

*Returns:* `data_table`, stored in the object

*Examples:*

```
\donttest{
data(soil_amp_network)
data(soil_amp)
# filter useless features to speed up the calculation
node_names <- unique(unlist(lapply(soil_amp_network, function(x){colnames(x$data_abund)})))
filter_soil_amp <- microeco::clone(soil_amp)
filter_soil_amp$otu_table <- filter_soil_amp$otu_table[node_names, ]
}
```

```

filter_soil_amp$tidy_dataset()
# obtain phylogenetic distance matrix
phylogenetic_distance <- as.matrix(cophenetic(filter_soil_amp$phylo_tree))
# choose the positive labels
t1 <- edge_node_distance$new(network_list = soil_amp_network,
  dis_matrix = phylogenetic_distance, label = "+")
}

```

**Method** `cal_diff()`: Differential test across networks.

*Usage:*

```

edge_node_distance$cal_diff(
  method = c("anova", "KW", "KW_dunn", "wilcox", "t.test")[1],
  ...
)

```

*Arguments:*

method default "anova"; see the following available options:

**'anova'** Duncan's multiple range test for anova

**'KW'** KW: Kruskal-Wallis Rank Sum Test for all groups ( $\geq 2$ )

**'KW\_dunn'** Dunn's Kruskal-Wallis Multiple Comparisons, see `dunnTest` function in `FSA` package

**'wilcox'** Wilcoxon Rank Sum and Signed Rank Tests for all paired groups

**'t.test'** Student's t-Test for all paired groups

... parameters passed to `cal_diff` function of `trans_alpha` class of `microeco` package.

*Returns:* `res_diff` in object. See the Return of `cal_diff` function in `trans_alpha` class of `microeco` package.

*Examples:*

```

\donttest{
t1$cal_diff(method = "wilcox")
}

```

**Method** `plot()`: Plot the distance.

*Usage:*

```

edge_node_distance$plot(...)

```

*Arguments:*

... parameters pass to `plot_alpha` function of `trans_alpha` class of `microeco` package.

*Returns:* `ggplot`.

*Examples:*

```

\donttest{
t1$plot(boxplot_add = "none", add_sig = TRUE)
}

```

**Method** `clone()`: The objects of this class are cloneable with this method.

*Usage:*

```

edge_node_distance$clone(deep = FALSE)

```

*Arguments:*

deep Whether to make a deep clone.



**Examples**

```

## -----
## Method `edge_node_distance$new`
## -----

data(soil_amp_network)
data(soil_amp)
# filter useless features to speed up the calculation
node_names <- unique(unlist(lapply(soil_amp_network, function(x){colnames(x$data_abund)})))
filter_soil_amp <- microeco::clone(soil_amp)
filter_soil_amp$otu_table <- filter_soil_amp$otu_table[node_names, ]
filter_soil_amp$tidy_dataset()
# obtain phylogenetic distance matrix
phylogenetic_distance <- as.matrix(cophenetic(filter_soil_amp$phylo_tree))
# choose the positive labels
t1 <- edge_node_distance$new(network_list = soil_amp_network,
  dis_matrix = phylogenetic_distance, label = "+")

## -----
## Method `edge_node_distance$cal_diff`
## -----

t1$cal_diff(method = "wilcox")

## -----
## Method `edge_node_distance$plot`
## -----

t1$plot(boxplot_add = "none", add_sig = TRUE)

```

---

edge\_tax\_comp

*Taxonomic sum of linked nodes in edges across networks*


---

**Description**

Taxonomic sum of linked nodes in edges across networks.

**Usage**

```
edge_tax_comp(network_list, taxrank = "Phylum", label = "+", rel = TRUE)
```

**Arguments**

network_list	a list with multiple networks; all the networks should be trans_network object created from trans_network class of microeco package.
taxrank	default "Phylum"; Which taxonomic level is used for the sum of nodes in edges.
label	default "+"; "+" or "-" or c("+", "-"); the edge label used for the selection of edges for the sum.
rel	default TRUE; TRUE represents using ratio, the denominator is the number of selected edges; FALSE represents the absolute number of the sum of edges.

**Value**

data.frame

**Examples**

```
data(soil_amp_network)
test <- edge_tax_comp(soil_amp_network)
# test is a microtable object
```

---

get\_edge\_table

*Get edge property table for each network*

---

**Description**

Get edge property table for each network in the list with multiple networks.

**Usage**

```
get_edge_table(network_list)
```

**Arguments**

network_list	a list with multiple networks; all the networks should be trans_network object created from trans_network class of microeco package.
--------------	--

**Value**

list, with res\_edge\_table in each network

**Examples**

```
data(soil_amp_network)
soil_amp_network <- get_edge_table(soil_amp_network)
```

---

get_node_table	<i>Get node property table for each network</i>
----------------	---

---

### Description

Get node property table for each network in the list with multiple networks.

### Usage

```
get_node_table(network_list, ...)
```

### Arguments

network_list	a list with multiple networks; all the networks should be trans_network object created from trans_network class of microeco package.
...	parameter passed to get_node_table function of trans_network class.

### Value

list, with res\_node\_table in each network

### Examples

```
data(soil_amp_network)
soil_amp_network <- get_node_table(soil_amp_network, node_roles = FALSE)
```

---

meconetcomp	<i>Introduction to meconetcomp package</i> ( <a href="https://github.com/ChiLiubio/meconetcomp">Rhrefhttps://github.com/ChiLiubio/meconetcomp</a> <a href="https://github.com/ChiLiubio/meconetcomp">https://github.com/ChiLiubio/meconetcomp</a> )
-------------	--

---

### Description

For the detailed tutorial on meconetcomp package, please follow the links:

Online tutorial website: [https://chiliubio.github.io/microeco\\_tutorial/meconetcomp-package.html](https://chiliubio.github.io/microeco_tutorial/meconetcomp-package.html)

Download tutorial: [https://github.com/ChiLiubio/microeco\\_tutorial/releases](https://github.com/ChiLiubio/microeco_tutorial/releases)

Please open the help document by using help function or by clicking the following links collected:

[cal\\_module](#)  
[cal\\_network\\_attr](#)  
[get\\_node\\_table](#)  
[get\\_edge\\_table](#)  
[node\\_comp](#)  
[edge\\_comp](#)  
[edge\\_node\\_distance](#)  
[edge\\_tax\\_comp](#)

[subset\\_network](#)  
[subnet\\_property](#)  
[robustness](#)  
[vulnerability](#)  
[cohesionclass](#)

To report bugs or discuss questions, please use Github Issues (<https://github.com/ChiLiubio/meconetcomp/issues>). Before creating a new issue, please read the guideline ([https://chiliubio.github.io/microeco\\_tutorial/notes.html#github-issues](https://chiliubio.github.io/microeco_tutorial/notes.html#github-issues)).

To cite meconetcomp package in publications, please run the following command to get the reference: `citation("meconetcomp")`

Reference:

Chi Liu, Chaonan Li, Yanqiong Jiang, Raymond Jianxiong Zeng, Minjie Yao, and Xiangzhen Li. 2023. A guide for comparing microbial co-occurrence networks. *iMeta*. 2(1): e71.

---

node_comp	<i>Generate a microtable object with node distributions across networks</i>
-----------	---

---

### Description

Generate a microtable object with node distributions across networks. Useful for the node information comparisons across different networks.

### Usage

```
node_comp(network_list, property = "name")
```

### Arguments

network_list	a list with multiple networks; all the networks should be trans_network object created from trans_network class of microeco package.
property	default "name"; a colname of res_node_table in each network; the default "name" represents using node presence/absence information in the otu_table of final output, in which 1 represents presence of the node in the corresponding network; For other options (such as degree), the results in the output otu_table are the actual values of res_node_table.

### Value

microtable object

### Examples

```
data(soil_amp_network)
test <- node_comp(soil_amp_network)
# test is a microtable object
```

---

robustness	<i>Calculate robustness across networks.</i>
------------	--

---

## Description

This class is a wrapper for robustness calculation and visualization.

## Methods

### Public methods:

- `robustness$new()`
- `robustness$plot()`
- `robustness$clone()`

### Method `new()`:

*Usage:*

```
robustness$new(
  network_list,
  remove_strategy = c("edge_rand", "edge_strong", "edge_weak", "node_rand", "node_hub",
    "node_degree_high", "node_degree_low")[1],
  remove_ratio = seq(0, 1, 0.1),
  measure = c("Eff", "Eigen", "Pcr")[1],
  run = 10
)
```

*Arguments:*

`network_list` a list with multiple networks; all the networks should be `trans_network` object created from `trans_network` class of `microeco` package.

`remove_strategy` default "edge\_rand";

"edge\_rand" edges are randomly removed.

"edge\_strong" edges are removed in decreasing order of weight.

"edge\_weak" edges are removed in increasing order of weight.

"node\_rand" nodes are removed randomly.

"node\_hub" node hubs are randomly removed. The hubs include network hubs and module hubs.

"node\_degree\_high" nodes are removed in decreasing order of degree.

"node\_degree\_low" nodes are removed in increasing order of degree.

`remove_ratio` default `seq(0, 1, 0.1)`.

`measure` default "Eff"; network robustness measures.

"Eff" network efficiency. The average efficiency of the network is defined:

$$Eff = \frac{1}{N(N-1)} \sum_{i \neq j \in G} \frac{1}{d(i, j)}$$

where  $N$  is the total number of nodes and  $d(i, j)$  is the shortest path between node  $i$  and node  $j$ . When the weight is found in the edge attributes,  $d(i, j)$  denotes the weighted

shortest path between node  $i$  and node  $j$ . For more details, please read the references <doi: 10.1007/s11704-016-6108-z> and <doi: 10.1038/s41598-020-60298-7>.

**"Eigen"** natural connectivity <doi: 10.1007/s11704-016-6108-z>. The natural connectivity can be regarded as an average eigenvalue that changes strictly monotonically with the addition or deletion of edges. It is defined:

$$\bar{\lambda} = \ln\left(\frac{1}{N} \sum_{i=1}^N e^{\lambda_i}\right)$$

where  $\lambda_i$  is the  $i$ th eigenvalue of the graph adjacency matrix. The larger the value of  $\bar{\lambda}$  is, the more robust the network is.

**"Pcr"** critical removal fraction of vertices (edges) for the disintegration of networks <doi: 10.1007/s11704-016-6108-z> <doi: 10.1103/PhysRevE.72.056130>. This is a robustness measure based on random graph theory. The critical fraction against random attacks is labeled as  $P_c^r$ . It is defined:

$$P_c^r = 1 - \frac{1}{\frac{\langle k^2 \rangle}{\langle k \rangle} - 1}$$

where  $\langle k \rangle$  is the average nodal degree of the original network, and  $\langle k^2 \rangle$  is the average of square of nodal degree.

run default 10. Replication number of simulation for the sampling method; Only available when `remove_strategy = "edge_rand", "node_rand" or "node_hub"`.

*Returns:* `res_table` and `res_summary`, stored in the object. The `res_table` is the original simulation result. The Mean and SD in `res_summary` come from the `res_table`.

*Examples:*

```
tmp <- robustness$new(soil_amp_network, remove_strategy = c("edge_rand"),
  measure = c("Eff"), run = 3, remove_ratio = c(0.1, 0.5, 0.9))
```

**Method** `plot()`: Plot the simulation results.

*Usage:*

```
robustness$plot(
  color_values = RColorBrewer::brewer.pal(8, "Dark2"),
  show_point = TRUE,
  point_size = 1,
  point_alpha = 0.6,
  show_errorbar = TRUE,
  errorbar_position = position_dodge(0),
  errorbar_size = 1,
  errorbar_width = 0.1,
  add_fitting = FALSE,
  ...
)
```

*Arguments:*

`color_values` colors used for presentation.

show\_point default TRUE; whether show the point.  
 point\_size default .3; point size value.  
 point\_alpha default .6; point alpha value.  
 show\_errorbar default TRUE; whether show the errorbar by using the SD result.  
 errorbar\_position default position\_dodge(0); Position adjustment, either as a string (such as "identity"), or the result of a call to a position adjustment function.  
 errorbar\_size default 1; errorbar size.  
 errorbar\_width default 0.1; errorbar width.  
 add\_fitting default FALSE; whether add fitted smooth line. FALSE denotes add line segment among points.  
 ... parameters pass to ggplot2::geom\_line (when add\_fitting = FALSE) or ggplot2::geom\_smooth (when add\_fitting = TRUE).

*Returns:* ggplot.

*Examples:*

```
\donttest{
tmp$plot(linewidth = 1)
}
```

**Method clone():** The objects of this class are cloneable with this method.

*Usage:*

```
robustness$clone(deep = FALSE)
```

*Arguments:*

deep Whether to make a deep clone.

## Examples

```
## -----
## Method `robustness$new`
## -----

tmp <- robustness$new(soil_amp_network, remove_strategy = c("edge_rand"),
  measure = c("Eff"), run = 3, remove_ratio = c(0.1, 0.5, 0.9))

## -----
## Method `robustness$plot`
## -----

tmp$plot(linewidth = 1)
```

---

soil_amp	<i>The soil_amp data</i>
----------	--------------------------

---

**Description**

The soil\_amp data is the 16S rRNA gene amplicon sequencing dataset of Chinese wetland soils.  
Reference: An et al. 2019 <doi:10.1016/j.geoderma.2018.09.035>; Liu et al. 2022 <10.1016/j.geoderma.2022.115866>

**Usage**

```
data(soil_amp)
```

---

soil_amp_network	<i>The soil_amp_network data</i>
------------------	----------------------------------

---

**Description**

The soil\_amp\_network data is a list storing three trans\_network objects created based on soil\_amp data. Three networks are created for IW, CW and TW groups, respectively.

**Usage**

```
data(soil_amp_network)
```

---

soil_measure_diversity	<i>The soil_measure_diversity data</i>
------------------------	--

---

**Description**

The soil\_measure\_diversity data is a table storing all the abiotic factors and functional diversity based on the metagenomic sequencing and MetaCyc pathway analysis.

**Usage**

```
data(soil_measure_diversity)
```



---

stool_met	<i>The stool_met data</i>
-----------	---------------------------

---

**Description**

The stool\_met data is the metagenomic species abundance dataset of stool samples selected from R ExperimentHub package. It has 198 samples, collected from the people with alcohol drinking habit, and 92 species.

**Usage**

```
data(stool_met)
```

---

subnet_property	<i>Calculate properties of sub-networks selected according to features in samples</i>
-----------------	---

---

**Description**

Extracting sub-network according to the presence of features in each sample across networks and calculate the sub-network properties.

**Usage**

```
subnet_property(network_list)
```

**Arguments**

network\_list a list with multiple networks; all the networks should be trans\_network object created from trans\_network class of microeco package.

**Value**

```
data.frame
```

**Examples**

```
data(soil_amp_network)
test <- subnet_property(soil_amp_network)
```

---

subset_network	<i>Extract subset of network according to the edge intersection of networks</i>
----------------	---

---

### Description

Extracting a network according to the edge intersection of networks.

### Usage

```
subset_network(network_list, venn = NULL, name = NULL)
```

### Arguments

network_list	a list with multiple networks; all the networks should be trans_network object created from trans_network class of microeco package.
venn	default NULL; a microtable object which must be converted by trans_comm function of trans_venn class.
name	default NULL; integer or character; must be a number or one of colnames of the otu_table in the input venn parameter.

### Value

trans\_network object, with only the extracted edges in the network

### Examples

```
data(soil_amp_network)
# first obtain edge distribution
tmp <- edge_comp(soil_amp_network)
# obtain edge intersection using trans_venn class
tmp1 <- microeco::trans_venn$new(tmp)
# convert intersection result to microtable object
tmp2 <- tmp1$trans_comm()
# extract the intersection of all the three networks ("IW", "TW" and "CW")
test <- subset_network(soil_amp_network, venn = tmp2, name = "IW&TW&CW")
# test is a trans_network object
```

---

vulnerability	<i>Calculate the vulnerability of each node for each network</i>
---------------	--

---

**Description**

The vulnerability of each node represents the influence of the node on the global efficiency of the network, i.e. the efficiency of network after removing the targeted node. For the detailed definition of global efficiency, please see the "Eff" option of measure parameter in [robustness](#) class.

**Usage**

```
vulnerability(network_list)
```

**Arguments**

`network_list` a list with multiple networks; all the networks should be `trans_network` object created from `trans_network` class of `microeco` package.

**Value**

data.frame

**Examples**

```
data(soil_amp_network)
vulnerability_table <- vulnerability(soil_amp_network)
```

# Index

## \* **Description**

meconetcomp, [11](#)

## \* **data.frame**

soil\_measure\_diversity, [16](#)

## \* **list**

soil\_amp\_network, [16](#)

## \* **microtable**

soil\_amp, [16](#)

stool\_met, [17](#)

cal\_module, [2](#), [11](#)

cal\_network\_attr, [3](#), [11](#)

cohesionclass, [4](#), [12](#)

edge\_comp, [6](#), [11](#)

edge\_node\_distance, [7](#), [11](#)

edge\_tax\_comp, [9](#), [11](#)

get\_edge\_table, [10](#), [11](#)

get\_node\_table, [11](#), [11](#)

meconetcomp, [11](#)

node\_comp, [11](#), [12](#)

robustness, [12](#), [13](#), [19](#)

soil\_amp, [16](#)

soil\_amp\_network, [16](#)

soil\_measure\_diversity, [16](#)

stool\_met, [17](#)

subnet\_property, [12](#), [17](#)

subset\_network, [12](#), [18](#)

vulnerability, [12](#), [19](#)