

# Package ‘riverconn’

August 6, 2022

**Type** Package

**Title** Common Fragmentation and Connectivity Indices for Riverscapes

**Version** 0.3.22

**Maintainer** Damiano Baldan <damiano.baldan91@gmail.com>

**Description** Indices for assessing riverscape fragmentation, including the Dendritic Connectivity Index, the Population Connectivity Index, the River Fragmentation Index, the Probability of Connectivity, and the Integral Index of connectivity. For a review, see Jumaní et al. (2020) <doi:10.1088/1748-9326/abcb37> and Baldan et al. (2022) <doi:10.1016/j.envsoft.2022.105470> Functions to calculate temporal indices improvement when fragmentation due to barriers is reduced are also included.

**License** MIT + file LICENSE

**Encoding** UTF-8

**Imports** doParallel, dplyr, foreach, igraph, magrittr, markdown, parallel, rlang, stats, tidyr, tidyselect, dodgr, reshape2

**Suggests** knitr, ggnetwork, ggplot2, viridis, rmarkdown

**RoxygenNote** 7.1.2

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Damiano Baldan [aut, cre] (<<https://orcid.org/0000-0001-9237-4883>>),  
David Cunillera-Montcusi [ctb]  
(<<https://orcid.org/0000-0001-8666-346X>>),  
Andrea Funk [ctb] (<<https://orcid.org/0000-0002-0568-1234>>)

**Repository** CRAN

**Date/Publication** 2022-08-06 14:00:07 UTC

## R topics documented:

B_ij_fun . . . . .	2
c_ij_fun . . . . .	3
d_index_calculation . . . . .	4
index_calculation . . . . .	6

set_graph_directionality . . . . .	9
t_index_calculation . . . . .	9
t_passability_sequencer . . . . .	11
t_weights_sequencer . . . . .	12

<b>Index</b>	<b>14</b>
--------------	-----------

---

B_ij_fun	<i>Calculates B<sub>ij</sub>: the functional contribution to dispersal probability I<sub>ij</sub></i>
----------	---

---

### Description

Calculates B<sub>ij</sub>: the functional contribution to dispersal probability I<sub>ij</sub>

### Usage

```
B_ij_fun(
  graph,
  field_B = "length",
  dir_distance_type = "symmetric",
  disp_type = "exponential",
  param_u,
  param_d,
  param
)
```

### Arguments

graph	an object of class igraph. Can be both directed or undirected.
field_B	the 'graph' edge attribute to be used to calculate the distance. Default is "length".
dir_distance_type	how directionality in B <sub>ij</sub> calculations is dealt with: "symmetric" (i.e. undirected graph) or "asymmetric" (i.e. directed graph). See details.
disp_type	the formula used to calculate the probabilities in the B <sub>ij</sub> matrix. Use "exponential" for exponential decay, or "threshold" for setting a distance threshold.
param_u	the upstream dispersal parameter. Must be a numeric value. Only used if dir_distance_type = "asymmetric". See details.
param_d	the downstream dispersal parameter. Must be a numeric value. Only used if dir_distance_type = "asymmetric". See details.
param	the dispersal parameter. Must be a numeric value. Only used if dir_distance_type = "symmetric". See details.

## Details

`dir_distance_type = "symmetric"` is to be used when the directionality of the river network is not relevant. The distance between reaches midpoints is calculated for each couple of reaches. `dir_distance_type = "asymmetric"` is to be used when the directionality is relevant. The distance between reaches midpoints is calculated for each couple of reaches and splitted between 'upstream travelled' distance and 'downstream travelled' distance

The 'param\_u', 'param\_d', and 'param' values are interpreted differently based on the formula used to relate distance and probability. When `disp_type = "exponential"`, those values are used as the base of the exponential dispersal kernel:  $B_{ij} = \text{param}^{d_{ij}}$ . When `disp_type = "threshold"`, those values are used to define the maximum dispersal length:  $B_{ij} = \text{ifelse}(d_{ij} < \text{param}, 1, 0)$ .

## Value

a square matrix of size `length(V(graph))` containing  $B_{ij}$  values. The matrix is organized with "from" nodes on the columns and "to" nodes on the rows

## Examples

```
library(igraph)
g <- igraph::graph_from_literal(1--2, 2--5, 3--4, 4--5, 6--7, 7--10, 8--9, 9--10,
5--11, 11--12, 10--13, 13--12, 12--14, 14--15, 15--16)
E(g)$id_dam <- c("1", NA, "2", "3", NA, "4", NA, "5", "6", NA, NA, NA, NA, "7", NA)
E(g)$type <- ifelse(is.na(E(g)$id_dam), "joint", "dam")
V(g)$length <- c(1, 1, 2, 3, 4, 1, 5, 1, 7, 7, 3, 2, 4, 5, 6, 9)
V(g)$HSI <- c(0.2, 0.1, 0.3, 0.4, 0.5, 0.5, 0.5, 0.6, 0.7, 0.8, 0.8, 0.8, 0.8, 0.8, 0.8)
V(g)$Id <- V(g)$name
E(g)$pass_u <- E(g)$pass_d <- ifelse(!is.na(E(g)$id_dam), 0.1, NA)
dist_mat <- B_ij_fun(g, param = 0.9)
```

---

c_ij_fun	<i>Calculates c_ij: the structural contribution to the dispersal probability I_ij</i>
----------	---

---

## Description

Calculates  $c_{ij}$ : the structural contribution to the dispersal probability  $I_{ij}$

## Usage

```
c_ij_fun(
  graph,
  dir_fragmentation_type = "symmetric",
  pass_confluence = 1,
  pass_u = "pass_u",
  pass_d = "pass_d"
)
```

**Arguments**

graph	an object of class <code>igraph</code> . Can be both directed or undirected.
dir_fragmentation_type	how directionality in <code>c_ij</code> calculations is dealt with: "symmetric" (i.e. undirected graph) or "asymmetric" (i.e. directed graph). See details.
pass_confluence	a value in the range [0,1] that defines the passability of confluences (default is 1).
pass_u	the 'graph' edge attribute to be used as upstream passability. Default is "pass_u".
pass_d	the 'graph' edge attribute to be used as downstream passability. Default is "pass_d".

**Details**

`dir_fragmentation_type = "symmetric"` is to be used when the directionality of the river network is not relevant. The equivalent passability for each barrier is calculated as the product of upstream and downstream passabilities. `dir_fragmentation_type = "asymmetric"` is to be used when the directionality is relevant. The equivalent passability of each barrier is calculated as a function of the path connecting each couple of reaches and depends on the direction of the path. Check the package vignette for more details.

**Value**

a square matrix of size `length(V(graph))` containing `c_ij` values. The matrix is organized with "from" nodes on the columns and "to" nodes on the rows

**Examples**

```
library(igraph)
g <- igraph::graph_from_literal(1--2, 2--5, 3--4, 4--5, 6--7, 7--10,
8--9, 9--10, 5--11, 11--12, 10--13, 13--12, 12--14, 14--15, 15--16)
E(g)$id_dam <- c("1", NA, "2", "3", NA, "4", NA, "5", "6", NA, NA, NA, NA, "7", NA)
E(g)$type <- ifelse(is.na(E(g)$id_dam), "joint", "dam")
V(g)$length <- c(1, 1, 2, 3, 4, 1, 5, 1, 7, 7, 3, 2, 4, 5, 6, 9)
V(g)$HSI <- c(0.2, 0.1, 0.3, 0.4, 0.5, 0.5, 0.5, 0.6, 0.7, 0.8, 0.8, 0.8, 0.8, 0.8, 0.8, 0.8)
V(g)$Id <- V(g)$name
E(g)$pass_u <- E(g)$pass_d <- ifelse(!is.na(E(g)$id_dam), 0.1, NA)
dist_mat <- c_ij_fun(g)
```

---

d_index_calculation	<i>Calculate Reach- and Catchment-scale index improvement for scenarios of barriers removal</i>
---------------------	---

---

**Description**

Calculate Reach- and Catchment-scale index improvement for scenarios of barriers removal

**Usage**

```
d_index_calculation(
    graph,
    ...,
    barriers_metadata,
    id_barrier = "id_barrier",
    pass_u_updated = "pass_u_updated",
    pass_d_updated = "pass_d_updated",
    mode = "leave_one_out",
    parallel = TRUE,
    ncores
)
```

**Arguments**

graph	an object of class 'igraph'. Can be both directed or undirected.
...	other arguments passed to the function 'index_calculation'
barriers_metadata	data.frame that must contain a column having the same name as the 'id_barrier' attribute of the graph, and two columns with the corresponding upstream and downstream improved passabilities (see 'pass_u_updated' and 'pass_d_updated' parameters).
id_barrier	graph edges attribute used to label barriers. Default is "id_barrier". Must be of type character.
pass_u_updated	field in barrier_metadata where updated value for upstream passability is stored (recommended values higher than the original passability).
pass_d_updated	field in barrier_metadata where updated value for downstream passability is stored (recommended values higher than the original passability).
mode	currently only "leave_one_out" is implemented.
parallel	logical value to flag if parallel option is to be used.
ncores	define how many cores are used in parallel processing. Active only when parallel = TRUE

**Details**

Setting `c_ij_flag = FALSE` (see `index_calculation` arguments) removes from the calculations the effect of barriers, i.e. the `c_ij` contribution is not used in the calculation of the index. Setting `B_ij_flag = FALSE` (see `index_calculation` arguments) removes from the calculations the effect of movement/dispersal, i.e. the `B_ij` contribution is not used in the calculation of the index. Note that it is not possible to set both `c_ij_flag = FALSE` and `B_ij_flag = FALSE`.

The setting `dir_distance_type = "symmetric"` (see `index_calculation` arguments) is to be used when the directionality of the river network is not relevant. The distance between reaches midpoints is calculated for each couple of reaches. The setting `dir_distance_type = "asymmetric"` (see `index_calculation` arguments) is to be used when the directionality is relevant. The distance between reaches midpoints is calculated for each couple of reaches and splitted between 'upstream travelled' distance and 'downstream travelled' distance

The 'param\_u', 'param\_d', and 'param' values are interpreted differently based on the formula used to relate distance and probability. When `disp_type = "exponential"` (see `index_calculation` arguments), those values are used as the base of the exponential dispersal kernel:  $B_{ij} = \text{param}^{d_{ij}}$ . When `disp_type = "threshold"` (see `index_calculation` arguments), those values are used to define the maximum dispersal length:  $B_{ij} = \text{ifelse}(d_{ij} < \text{param}, 1, 0)$ .

### Value

returns a data.frame containing the percent improvement of the index for each barrier present in the 'barriers\_metadata' variable. If `index_type = "full"` (see `index_calculation` arguments), the data.frame is organized by 'id\_barrier'. If `index_type = "reach"` (see `index_calculation` arguments), the data.frame is organized by 'id\_barrier' and 'name'. In both cases, both numerator and denominator used in the index calculations are reported in the columns 'num' and 'den'. The column 'd\_index' contains the relative index improvement when each barrier is removed.

### Examples

```
library(igraph)
library(igraph)
g <- igraph::graph_from_literal(1--2, 2--4, 3--2, 4--6, 6--7, 5--6, 7--8, 9--5, 10--5 )
E(g)$id_dam <- c(NA, NA, "1", NA, NA, "2", NA, NA, NA)
E(g)$type <- ifelse(is.na(E(g)$id_barrier), "joint", "dam")
V(g)$length <- c(1, 1, 2, 3, 4, 1, 5, 1, 2, 1)
V(g)$Id <- V(g)$name
E(g)$pass_u <- E(g)$pass_d <- ifelse(!is.na(E(g)$id_dam), 0.1, NA)
dams_metadata <- data.frame("id_dam" = c("1", "2"),
"pass_u_updated" = c(1, 1), "pass_d_updated" = c(1, 1))
d_index <- d_index_calculation(g, barriers_metadata = dams_metadata,
id_barrier = "id_dam", parallel = FALSE, param = 0.6)
```

---

index\_calculation

*Reach- and Catchment-scale indices of connectivity*

---

### Description

Reach- and Catchment-scale indices of connectivity

### Usage

```
index_calculation(
  graph,
  weight = "length",
  nodes_id = "name",
  index_type = "full",
  index_mode = "to",
  c_ij_flag = TRUE,
```

```

    B_ij_flag = TRUE,
    dir_fragmentation_type = "symmetric",
    pass_confluence = 1,
    pass_u = "pass_u",
    pass_d = "pass_d",
    field_B = "length",
    dir_distance_type = "symmetric",
    disp_type = "exponential",
    param_u,
    param_d,
    param
)

```

### Arguments

graph	an object of class <code>igraph</code> . Can be both directed or undirected.
weight	graph vertex attribute used to assign weights to the reaches (nodes/vertices). Should not be also an edge attribute. Default is "length".
nodes_id	graph vertex attribute used to univoquely label reaches (nodes/vertices). Should not be also an edge attribute. Default is "name". The graph attribute must be a character vector. Used to label the results when <code>index_type = "reach"</code>
index_type	indicates if the index should be calculated for the whole catchment ( <code>index_type = "full"</code> ), for each reach ( <code>index_type = "reach"</code> ), or for each barrier ( <code>index_type = "sum"</code> )
index_mode	indicates if reach index should be calculated based on inbound links ("to") or outbound links ("from"). Only active when <code>index_type = "reach"</code> .
c_ij_flag	include the presence of barriers in the calculations ( <code>c_ij</code> term).
B_ij_flag	include dispersal/movement among reaches in the calculations ( <code>B_ij</code> term).
dir_fragmentation_type	how directionality in <code>c_ij</code> calculations is dealt with: "symmetric" (i.e. undirected graph) or "asymmetric" (i.e. directed graph). See details below.
pass_confluence	a value in the range [0,1] that defines the passability of confluences (default is 1).
pass_u	the 'graph' edge attribute to be used as upstream passability. Default is "pass_u".
pass_d	the 'graph' edge attribute to be used as downstream passability. Default is "pass_d".
field_B	the 'graph' vertex attribute to be used to calculate the distance. Should not be also an edge attribute. Default is "length".
dir_distance_type	how directionality in <code>B_ij</code> calculations is dealt with: "symmetric" (i.e. undirected graph) or "asymmetric" (i.e. directed graph). See details below.
disp_type	the formula used to calculate the probabilities in the <code>B_ij</code> matrix. Use "exponential" for exponential decay, or "threshold" for setting a distance threshold.

param_u	upstream dispersal parameter. Must be a numeric value. Only used if dir_distance_type = "asymmetric". See details below.
param_d	downstream dispersal parameter. Must be a numeric value. Only used if dir_distance_type = "asymmetric". See below for details.
param	dispersal parameter. Must be a numeric value. Only used if dir_distance_type = "symmetric". See details below.

### Details

Setting `c_ij_flag = FALSE` removes from the calculations the effect of barriers, i.e. the `c_ij` contribution is not used in the calculation of the index. Setting `B_ij_flag = FALSE` removes from the calculations the effect of movement/dispersal, i.e. the `B_ij` contribution is not used in the calculation of the index. Note that it is not possible to set both `c_ij_flag = FALSE` and `B_ij_flag = FALSE`.

The setting `dir_distance_type = "symmetric"` is to be used when the directionality of the river network is not relevant. The distance between reaches midpoints is calculated for each couple of reaches. The setting `dir_distance_type = "asymmetric"` is to be used when the directionality is relevant. The distance between reaches midpoints is calculated for each couple of reaches and splitted between 'upstream travelled' distance and 'downstream travelled' distance

The 'param\_u', 'param\_d', and 'param' values are interpreted differently based on the formula used to relate distance and probability. When `disp_type = "exponential"`, those values are used as the base of the exponential dispersal kernel:  $B_{ij} = \text{param}^{d_{ij}}$ . When `disp_type = "threshold"`, those values are used to define the maximum dispersal length:  $B_{ij} = \text{ifelse}(d_{ij} < \text{param}, 1, 0)$ .

### Value

If `index_type = "full"`, returns a numeric value with the index value (column 'index'). if `index_type = c("reach", "sum")`, returns a data frame with the index value (column 'index') for each reach (the field specified in 'nodes\_id' is used for reach identification in the data frame). In both cases, both numerator and denominator used in the index calculations are reported in the columns 'num' and 'den'.

### Examples

```
library(igraph)
g <- igraph::graph_from_literal(1--2, 2--5, 3--4, 4--5, 6--7,
7--10, 8--9, 9--10, 5--11, 11--12, 10--13, 13--12, 12--14, 14--15, 15--16)
E(g)$id_dam <- c("1", NA, "2", "3", NA, "4", NA, "5", "6", NA, NA, NA, NA, "7", NA)
E(g)$type <- ifelse(is.na(E(g)$id_dam), "joint", "dam")
V(g)$length <- c(1, 1, 2, 3, 4, 1, 5, 1, 7, 7, 3, 2, 4, 5, 6, 9)
V(g)$HSI <- c(0.2, 0.1, 0.3, 0.4, 0.5, 0.5, 0.5, 0.6, 0.7, 0.8, 0.8, 0.8, 0.8, 0.8, 0.8, 0.8)
V(g)$Id <- V(g)$name
E(g)$pass_u <- E(g)$pass_d <- ifelse(!is.na(E(g)$id_dam), 0.1, NA)
index <- index_calculation(g, param = 0.9)
```



---

`set_graph_directionality`*Create directed river graph based on outlet flag*

---

**Description**

The input graph can be either directed or undirected. If directed, then it is made undirected before directionality is assigned.

**Usage**

```
set_graph_directionality(graph, field_name = "name", outlet_name)
```

**Arguments**

<code>graph</code>	an 'igraph' object representing a river structure where reaches are nodes and confluences (or fragmentation items) are links.
<code>field_name</code>	a character value that flags the vertices attribute used to designate the outlet. Each vertex must have a unique value for this field.
<code>outlet_name</code>	a character value corresponding to the 'field_name' attribute

**Value**

an object of class 'igraph' containing a directed graph.

**Examples**

```
library(igraph)
g <- igraph::graph_from_literal(1-2, 2-4, 3-2, 4-6, 6-7, 5-6, 7-8, 9-5, 10-5 )
g1 <- set_graph_directionality(g, field_name = "name", "8")
```

---

`t_index_calculation`     *Calculates time-dependent index when nodes weights or barriers passability are changing*

---

**Description**

Calculates time-dependent index when nodes weights or barriers passability are changing

**Usage**

```
t_index_calculation(
  graph = graph,
  ...,
  barriers_metadata,
  id_barrier = "id_barrier",
  year = "year",
  pass_u = "pass_u",
  pass_d = "pass_d",
  weights_metadata,
  weight = "length",
  nodes_id = "name",
  parallel = TRUE,
  ncores
)
```

**Arguments**

<code>graph</code>	an object of class <code>igraph</code> . Can be both directed or undirected.
<code>...</code>	other arguments passed to the function <code>index_calculation</code>
<code>barriers_metadata</code>	data.frame that must contain a column having the same name as the <code>'id_barrier'</code> attribute of the <code>graph</code> , and two columns with the corresponding upstream and downstream improved passabilities (see <code>pass_u</code> and <code>pass_d</code> ), and a column with the year passability was changed. This data frame can be obtained from easily-formatted data with the function <code>t_passability_sequencer</code> .
<code>id_barrier</code>	graph edges attribute used to label barriers. Default is <code>"id_barrier"</code> . It should be present in the <code>'barriers metadata'</code> input as well.
<code>year</code>	field of the <code>'barriers metadata'</code> where temporal information on the changes in passability is stored.
<code>pass_u</code>	field of the <code>'barriers metadata'</code> where temporal-dependent upstream passability is stored.
<code>pass_d</code>	field of the <code>'barriers metadata'</code> where temporal-dependent downstream passability is stored.
<code>weights_metadata</code>	data.frame that must contain a column having the same name as the <code>'nodes_id'</code> attribute of the <code>graph</code> , a column with the corresponding weight information (see <code>'weight'</code> parameter), and a column with the year weight was changed. This data frame can be obtained from easily-formatted data with the function <code>t_weight_sequencer</code> .
<code>weight</code>	param weight graph vertex attribute used to assign weights to the reaches (nodes). Default is <code>"length"</code> .
<code>nodes_id</code>	graph vertex attribute used to uniquely label reaches (nodes). Default is <code>"name"</code> .
<code>parallel</code>	logical value to flag if parallel option is to be used.
<code>ncores</code>	define how many cores are used in parallel processing. Active only when <code>parallel = TRUE</code>

**Value**

a data.frame with a 'year' field and related connectivity index. If index\_type = "reach", the data.frame is organized by 'year' and 'name'.

**Examples**

```
library(igraph)
g <- igraph::graph_from_literal(1--2, 2--4, 3--2, 4--6, 6--5)
E(g)$id_barrier <- c(NA, NA, "1", NA, NA)
E(g)$type <- ifelse(is.na(E(g)$id_barrier), "joint", "dam")
V(g)$length <- c(1, 1, 2, 3, 4,5)
V(g)$Id <- V(g)$name
E(g)$pass_u <- E(g)$pass_d <- ifelse(!is.na(E(g)$id_barrier),0.1,NA)
barriers_data <- data.frame("id_barrier" = c("1"),
  "year_c" = 2000, "pass_c_u" = 0.1, "pass_c_d" = 0.4)
seq_ops <- c("c")
barriers_metadata <- t_passability_sequencer(barriers_data, seq_ops)
weights_dataframe <- data.frame("name" = seq(1,6) %>% as.character,
  "length_1999" = c(1, 1, 2, 3, 4,5))
weights_metadata <- t_weights_sequencer(weights_dataframe, weight = "length")
t_index <- t_index_calculation(g, barriers_metadata = barriers_metadata,
  weights_metadata = weights_metadata, weight = "length", parallel = FALSE, B_ij_flag = FALSE)
```

---

t\_passability\_sequencer

*Create the time-dependent metadata for barriers*

---

**Description**

Create the time-dependent metadata for barriers

**Usage**

```
t_passability_sequencer(passability_information, seq_ops)
```

**Arguments**

passability\_information

a data frame in wide format. Must contain an 'id\_barrier' column. Each change in passability is listed in a group of 3 columns: 'year\_op', 'pass\_op\_u', and 'pass\_op\_d', listing the year the operation (op) took place, and the related upstream and downstream passabilities. In case the passability did not change, a NA value should be used. See details.

seq\_ops

A character vector with the temporal sequence of operations. It should contain all the operation strings in the 'passability\_information' data frame.

**Details**

This function is meant to help processing data the way they can be obtained from a database, or the way they are stored in a spreadsheet. The substring 'op' in the fields 'year\_op', 'pass\_op\_u', and 'pass\_op\_d' is used to identify each operation and to relate it to the relative passability parameters. For example, c can be used for construction, and fp for the implementation of a fish pass. In this case, passability\_information will have the fields 'year\_c', 'pass\_c\_u', and 'pass\_c\_d', 'year\_fp', 'pass\_fp\_u', and 'pass\_fp\_d'. Then, the input seq\_ops = c("c", "fp"), meaning that first the operation named 'c' occurred, and then the operation named 'fp' occurred.

**Value**

a dataframe in a long format that can be used as input to the tDCI function.

**Examples**

```
barriers_data <- data.frame("id_barrier" = c("1", "2"),
  "year_c" = c(1950, 1990), "pass_c_u" = c(0.1, 0.1), "pass_c_d" = c(0.4, 0.4),
  "year_fp" = c(2000, 2010), "pass_fp_u" = c(0.5, 0.5), "pass_fp_d" = c(0.8, 0.8))
seq_ops <- c("c", "fp")
t_metadata <- t_passability_sequencer(barriers_data, seq_ops)
```

---

t\_weights\_sequencer    *Create the time-dependent weights data*

---

**Description**

Create the time-dependent weights data

**Usage**

```
t_weights_sequencer(weights_information, weight = "length", nodes_id = "name")
```

**Arguments**

weights_information	a data.frame that must contain a 'nodes_id' column and several 'weight' columns. Weight columns are named with the string contained in the 'weight' input and the relative year (4 digits format), separated by an underscore (e.g. when weight = "length", the names of the 'weight' columns will be: 'weight_1990', 'weight_2000', 'weight_2020', etc.).
weight	a character object containing the label of the columns whose weight change with time
nodes_id	a character object containing the label of the columns that uniquely identify reaches.

**Value**

a data frame with columns 'name', 'year', and 'weight' to be used in the function `t_index_calculation`

**Examples**

```
weights_dataframe <- data.frame("id" = c("1", "2", "3", "4", "5"),  
  "weight_1900" = c(10, 15, 100, 50, 40),  
  "weight_1950" = c(11, 16, 90, 55, 45),  
  "weight_2000" = c(13, 19, 80, 49, 44))  
weights_metadata <- t_weights_sequencer(weights_dataframe, weight = "weight", nodes_id = "id")
```

# Index

[B\\_ij\\_fun](#), [2](#)

[c\\_ij\\_fun](#), [3](#)

[d\\_index\\_calculation](#), [4](#)

[index\\_calculation](#), [6](#)

[set\\_graph\\_directionality](#), [9](#)

[t\\_index\\_calculation](#), [9](#)

[t\\_passability\\_sequencer](#), [11](#)

[t\\_weights\\_sequencer](#), [12](#)