Package 'riverconn'

January 16, 2023

Type Package

Title Common Fragmentation and Connectivity Indices for Riverscapes

Version 0.3.25

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 Jumani 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, parallel, rlang, stats, tidyr, tidyselect, dodgr, reshape2

Suggests knitr, ggnetwork, ggplot2, viridis, rmarkdown

RoxygenNote 7.2.3

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 2023-01-16 18:20:08 UTC

R topics documented:

`B_ij_fun` *Calculates B_ij: the functional contribution to dispersal probability I_ij*

Description

Calculates B_ij: the functional contribution to dispersal probability I_ij

Usage

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

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_ij 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_ij matrix. Use `"exponential"` for exponential decay, `"threshold"` for setting a distance threshold, or `"leptokurtic"` for leptokurtic dispersal.
`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.
`param_l`	the parameters for the leptokurtic dispersal mode. Must be a numeric vector of the type `c(sigma_stat, sigma_mob, p)`. See details below.

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. When disp_type ="leptokurtic" is selected, symmetric dispersal is assumed.

The 'param_u', 'param_d', and 'param' values are interpreted differently based on the formula used to relate distance (d_ij) and probability (B_ij). When disp_type ="exponential", those values are used as the base of the exponential dispersal kernel: B_ij = param^d_ij. When disp_type ="threshold", those values are used to define the maximum dispersal length: B_ij = ifelse(d_ij < param, 1, 0).

When disp_type ="leptokurtic" is selected, a leptokurtic dispersal kernel is used to calculate B_ij. A leptokurtic dispersal kernel is a mixture of two zero-centered gaussian distributions with standard deviations sigma_stat (static part of the population), and sigma_mob (mobile part of the population). The probability of dispersal is calculated as: B_ij = p F(0, sigma_stat, d_ij) + (1-p) F(0, sigma_mob, d_ij) where F is the upper tail of the gaussian cumulative density function.

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)$HSI <- c(0.2, 0.1, 0.3, 0.4, 0.5, 0.5, 0.5, 0.6, 0.7, 0.8, 0.8, 0.8, 0.8, 0.8, 0.8, 0.8)
V(g)$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 — *Calculates c_ij: the structural contribution to the dispersal probability I_ij*

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 igraph. Can be both directed or undirected.
- `dir_fragmentation_type` how directionality in c_ij 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` *Calculate Reach- and Catchment-scale index improvement for scenarios of barriers removal*

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 charachter.

`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` currentlym 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 = param^d_ij. When `disp_type ="threshold"` (see index_calculation arguments), those values are used to define the maximum dispersal length: B_ij = ifelse(d_ij < 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.

References

Baldan, D., Cunillera-Montcusí, D., Funk, A., & Hein, T. (2022). Introducing 'riverconn': an R package to assess river connectivity indices. Environmental Modelling & Software, 156, 105470.

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)$id_dam), "joint", "dam")
V(g)$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,
  param_l
)
```

Arguments

graph	an object of class igraph. 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 index_type = "reach"
index_type	indicates if the index should be calculated for the whole catchment (index_type = "full"), for each reach (index_type = "reach"), or for each barrier (index_type = "sum")
index_mode	indicates if reach index should be calculated based on inbound links ("to") or outbound links ("from"). Only active when index_type = "reach".
c_ij_flag	include the presence of barriers in the calculations (c_ij term).
B_ij_flag	include dispersal/movement among reaches in the calculations (B_ij term).

dir_fragmentation_type
: how directionality in c_ij 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 B_ij 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_ij matrix. Use "exponential" for exponential decay, "threshold" for setting a distance threshold, or "leptokurtic" for leptokurtic dispersal.

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.

param_l
: the parameters for the leptokurtic dispersal mode. Must be a numeric vector of the type c(sigma_stat, sigma_mob, p). 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. When disp_type ="leptokurtic" is selected, symmetric dispersal is assumed.

The 'param_u', 'param_d', and 'param' values are interpreted differently based on the formula used to relate distance (d_ij) and probability (B_ij). When disp_type ="exponential", those values are used as the base of the exponential dispersal kernel: B_ij = param^d_ij. When disp_type ="threshold", those values are used to define the maximum dispersal length: B_ij = ifelse(d_ij < param, 1, 0).

When disp_type ="leptokurtic" is selected, a leptokurtic dispersal kernel is used to calculate B_ij. A leptokurtic dispersal kernel is a mixture of two zero-centered gaussian distributions with standard deviations `sigma_stat` (static part of the population), and `sigma_mob` (mobile part of the population). The probability of dispersal is calculated as: B_ij = p F(0, sigma_stat, d_ij) + (1-p) F(0, sigma_mob, d_ij) where F is the upper tail of the gaussian cumulative density function.

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'.

References

Baldan, D., Cunillera-Montcusí, D., Funk, A., & Hein, T. (2022). Introducing 'riverconn': an R package to assess river connectivity indices. Environmental Modelling & Software, 156, 105470.

Jumani, S., Deitch, M. J., Kaplan, D., Anderson, E. P., Krishnaswamy, J., Lecours, V., & Whiles, M. R. (2020). River fragmentation and flow alteration metrics: a review of methods and directions for future research. Environmental Research Letters, 15(12), 123009.

Radinger, J., & Wolter, C. (2014). Patterns and predictors of fish dispersal in rivers. Fish and fisheries, 15(3), 456-473.

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

graph — an 'igraph' object representing a river structure where reaches are nodes and confluences (or fragmentation items) are links.

field_name — a character value that flags the vertices attribute used to designate the outlet. Each vertex must have an unique value for this field.

outlet_name — 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

`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 and pass_d), and a column with the year passability was changed. This data frame can be obtained from easily-formatted data with the function `t_passability_sequencer`.

`id_barrier` — graph edges attribute used to label barriers. Default is `"id_barrier"`. It should be present in the 'barriers metadata' input as well.

`year` — field of the 'barriers metadata' where temporal information on the changes in passability is stored.

`pass_u` — field of the 'barriers metadata' where temporal-dependent upstream passability is stored.

`pass_d` — field of the 'barriers metadata' where temporal-dependent downstream passability is stored.

`weights_metadata` — data.frame that must contain a column having the same name as the 'nodes_id' attribute of the graph, a column with he corresponding weight information (see 'weight' parameter), and a column with the year weight was changed. This data frame can be obtained from easily-formatted data with the function `t_weight_sequencer`.

`weight` — param weight graph vertex attribute used to assign weights to the reaches (nodes). Default is `"length"`.

`nodes_id` — graph vertex attribute used to uniquely label reaches (nodes). Default is `"name"`.

`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`

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'.

References

Baldan, D., Cunillera-Montcusí, D., Funk, A., & Hein, T. (2022). Introducing 'riverconn': an R package to assess river connectivity indices. Environmental Modelling & Software, 156, 105470.

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 charachter 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