Package 'toporanga'

July 22, 2025

Type Package

Title Topological Sort-Based Hierarchy Inference

Version 1.0.0

Depends R (>= 3.5.0)

License GPL-3

Description Deciphering hierarchy of agents exhibiting observable dominance events is a crucial problem in several disciplines, in particular in behavioural analysis of social animals, but also in social sciences and game theory. This package implements an inference approach based on graph theory, namely to extract the optimal acyclic subset of a weighted graph of dominance; this allows for hierarchy estimation through topological sorting. The package also contains infrastructure to investigate partially defined hierarchies and hierarchy dynamics.

BugReports https://gitlab.com/mbq/toporanga/-/issues

URL https://gitlab.com/mbq/toporanga

Encoding UTF-8

RoxygenNote 7.3.2

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation yes

Author Miron B. Kursa [aut, cre] (ORCID:
<https://orcid.org/0000-0001-7672-648X>)

Maintainer Miron B. Kursa <m@mbq.me>

Repository CRAN

Date/Publication 2025-07-16 17:20:02 UTC

Contents

all_toposorts 2
arrange 3
components 3

cumulative_epochs 4
diffuse 4
dot 5
EcoHAB 6
marginise 6
normalise 7
parameters 7
sample_toposort 8
tally_from_matrix 8
tally_log 9
toporanga 10
window_epochs 11
zero_opposition 12

Index **13**

`all_toposorts` *Extract all possible toposorts of a graph*

Description

Some toporanga graphs can be topologically sorted in many ways, but `arrange` will generate only one, random order. This function allows to list all of them. Use `sample_toposort` to fairly sample a single toposort.

Usage

```
all_toposorts(x, ..., limit = Inf)
```

Arguments

`x`	a `toporanga_graph` object.
`...`	ignored.
`limit`	maximal number of permutations returned; function errors when exhausted.

Value

A list of possible orderings, in a form of vectors of agent IDs.

arrange *Arrange agents into hierarchy*

Description

Estimate agent hierarchy from a tally by assembling edges in order of decreasing weight and skipping cycle-creating ones. The effect is a directed acyclic graph that can be topologically-sorted into an agent order, though this order may not be unique depending on how well the latent hierarchy is sampled and whether it even exists.

Usage

```
arrange(tally)
```

Arguments

tally	a toporanga tally object.

Value

A toporanga_graph object, representing the established graph.

Note

The method may be stochastic if there are ties in weights.

Examples

```
data(EcoHAB)
arrange(tally_log(EcoHAB$winner,EcoHAB$loser))
```

components *Extract components of a tally*

Description

Decomposes the tally into stronly connected components.

Usage

```
components(tally)
```

Arguments

tally	a toporanga_tally object.

Value

A vector of component IDs for each agent, named with agent IDs. Components are integers from 1 up to the component count, which is at most the number of agents. Hence, when the tally is fully strongly connected, the result is a vector of ones.

---

cumulative_epochs *Cumulative epoch generator*

Description

Splits the event log into cumulative epochs, i.e., starting from the start up to selected number of final points. Returns an object which has to be fed into `epochs` argument of `toporanga` function.

Usage

```
cumulative_epochs(n, t)
```

Arguments

`n`	number of epochs. Silently capped to the number of events, if larger.
`t`	optional event times, vector of a same length and order as the event log given to `toporanga` function. When given, epochs will be generated based on equal-time breaks not equal-event-number breaks.

Value

A special function that can be passed to the `epochs` argument of `toporanga` function.

Note

If time is not given, event log must be sorted; otherwise, epochs will not make any sense.

---

diffuse *Diffuse the weights over indirect links.*

Description

Uses the Schulze / widest path approach to re-express indirect domination into direct one. For instance, if $A \to B$ and $B \to C$ both have weight 10, but $A \to C$ has weight 3, it is gonna be corrected to 10 by this function.

Usage

```
diffuse(tally, blend)
```

Arguments

tally	a toporanga_tally object.
blend	a numerical factor to limit the effect of diffusion; weight of indirect path is multiplied by the given factor, while direct weights are not changed. When set to 1, function behaves the same as blend was not given at all, when given 0, whole diffuse returns tally unchanged.

Value

A modified toporanga_tally object.

---

dot *Export dominance graph into Graphviz dot format*

Description

Produces a string vector with the dot code that can be used to plot dominance graph with Graphviz or transfer topology to other software.

Usage

```
dot(x, con, ..., reduce = TRUE)
```

Arguments

x	a toporanga_graph to be exported.
con	if given, code is pushed to the given connection instead of returned; this can simply be a file name.
...	ignored.
reduce	if TRUE, a transitive reduction is applied to the graph, making it retain the order topology but removing all redundant edges.

Value

Graphviz dot code of the graph, invisibly when con is given.

EcoHAB *Example EcoHAB data*

Description

Mice dominance events recorded by the Eco-HAB system. Covers a group of 10 mice and 6575 events recorded over a span of almost 5 days, corresponding to followings in the system. Contains three columns: `winner`, ID of winning mouse, `loser`, ID of a losing mice, `t` time of event occurrence, in seconds since the first event.

Usage

```
data(EcoHAB)
```

Format

An object of class `data.frame` with 6575 rows and 3 columns.

marginise *Convert tally weights into margin weights*

Description

When we have $w(A \to B) = a$ and $w(B \to A) = b < a$, this function will change $w(A \to B)$ to $a - b$ and $w(B \to A)$ to 0.

Usage

```
marginise(tally)
```

Arguments

`tally`	a `toporanga_tally` object in which weights are going to be converted to margins.

Value

A modified `toporanga_tally` object.

normalise *Normalise tally weights*

Description

When we have $w(A \to B) = a$ and $w(B \to A) = b$, this function will normalise both to sum to 1, i.e. divide by $a + b$. This way dominance matters more than engagement in conflicts between particular agents; in particular, an agent that frequently loses in typical events but always dominate over the otherwise dominating agent will come on top with normalised weights, while remain near the bottom otherwise.

Usage

```
normalise(tally)
```

Arguments

tally	a toporanga_tally object in which weights are going to be normalised.

Value

A modified toporanga_tally object.

parameters *Parameters of agents from a graph*

Description

Pull numerical descriptors of a toporanga graph.

Usage

```
parameters(g)
```

Arguments

g	toporanga_graph object generated from tally.

Value

A `data.frame` object. Each of its rows corresponds to one agent. This agent's ID is listed in the `Agent` column. `Superiors` counts the number of agents which are upstream from a given agent, while `Subordinates` counts those downstream. This numbers correspond to a rank when when there is only one unique toposort of the dominance graph, and provide a graceful degradation if not. In particular they do not depend on random seed or the default toposort stored in the graph object. Thus, subordinate count is a recommended outcome if a single dominance score of an agent is desired. The `Order` column notes the aforementioned default toposort, which can be re-sampled with the `sample_toposort`. Additional columns may appear depending on meta-data in the graph object, in particular epoch parameters when using `epoch` argument of `toporanga` function.

`sample_toposort` *Switch default toposorts of a graph to a random one*

Description

Some toporanga graphs can be topologically sorted in many ways, but `arrange` will generate only one, random order. This function allows one to switch it to an another, randomly sampled one. Use `all_toposorts` to calculate all possible orderings.

Usage

```
sample_toposort(x)
```

Arguments

`x`	a toporanga graph.

Value

The same graph, with a fairly re-sampled default order.

`tally_from_matrix` *Create tally from a raw matrix*

Description

Creates a `toporanga_tally` objects from a matrix representing counts of domination events between pairs of agents.

Usage

```
tally_from_matrix(x)
```

Arguments

x	matrix to convert. It must be square, have zero diagonal, non-negative values and identical column and row names or lack of both ("1".."n" is gonna be inferred in this case).

Value

`toporanga_tally` object.

Note

If in doubt, use `tally_log` instead of this function.

`tally_log` *Create tally from an event log*

Description

Collects a log of domination events into a `toporanga_tally` object which can be fed to other functions in the package.

Usage

```
tally_log(winning, opposing, weight, ..., agents)
```

Arguments

`winning`	vector of IDs of winning agents.
`opposing`	vector of IDs of opposing agents.
`weight`	optional weight of the event; set to 1 for each event if not given.
`...`	ignored.
`agents`	optional vector ofg IDs of all agents; allows for ghost agents that are in the set but are not involved in any event.

Value

A `toporanga_tally` object.

toporanga *A catch-all function to access package functionality in one call*

Description

The general pipeline of toporanga is to tally the domination events, optionally modify the tally by diffusing event weights and/or re-scoring reciprocal weights, arrange agents into an order represented by an acyclic graph, finally to reduce it to an order or a dominance score. This function collects all of this into a single call.

Usage

```
toporanga(
  winning,
  opposing,
  weight,
  ...,
  agents,
  diffuse = FALSE,
  reciprocals = c(),
  output = c("parameters", "graph", "order_sample", "subordinate_count"),
  epochs
)
```

Arguments

winning	vector of IDs of winning agents.
opposing	vector of IDs of opposing agents.
weight	optional weight of the event; set to 1 for each event if not given.
...	ignored.
agents	optional vector of IDs of all agents; allows for ghost agents that are in the set but are not involved in any event.
diffuse	if TRUE, weight diffusion is applied. See diffuse for more details.
reciprocals	a vector of commands applied to the tally. "keep" does nothing; "marginise" reduces weights into weight margins, see marginise, "zero" zeroes opposing weights, see zero_opposition, "normalise" normalises weights within conflicts, see normalise; "diffuse" applies diffuse, allows to manage order in which diffuse is mixed with other modifiers; cannot be mixed with diffuse=TRUE. Normally, diffuse is applied first.
output	specifies what to return from the function. "parameters" returns the some agent parameters, see parameters; "graph" a toporanga_graph object; "order_sample" an order of agents; in case multiple is possible, a single random sample; "subordinate_count" a count of subordinate agents for each agent, which can be interpreted as a dominance score.

epochs — optional driver for applying analysis on subsets of events. See cumulative_epochs and window_epochs for more information. If given, changes the output to an aggregated version of whatever was selected with output.

Value

A result of the caclulation in a form specified by the output argument, aggregated over epochs is epochs argument is given.

Examples

```
data(EcoHAB)
toporanga(EcoHAB$winner,EcoHAB$loser)
```

---

window_epochs *Moving window epoch generator*

Description

Splits the event log into cumulative epochs, i.e., starting from the start up to selected number of final points. Returns an object which has to be fed into epochs argument of toporanga function.

Usage

```
window_epochs(n, window, t)
```

Arguments

n	number of epochs. Silently capped to the number of events, if larger. Inferred from window if missing.
window	size of the moving window; if t is given, in the units of time, otherwise in event count. Set to range/n if not given. Larger windows are just made overlapping.
t	optional event times, vector of a same length and order as the event log given to toporanga function. When given, epochs will be generated based on equal-time breaks not equal-event-number breaks.

Value

A special function that can be passed to the epochs argument of toporanga function.

Note

If time is not given, event log must be sorted; otherwise, epochs will not make any sense.

`zero_opposition` *Zero the opposing tally weights*

Description

When we have $w(A \to B) = a$ and $w(B \to A) = b < a$, this function will retain $w(A \to B) = a$ and change $w(B \to A)$ into 0.

Usage

```
zero_opposition(tally, quench = FALSE)
```

Arguments

`tally`	a `toporanga_tally` object in which opposing weights are going to be zeroed.
`quench`	whether to remove tied-up weights.

Value

A modified `toporanga_tally` object.

Index

∗ **datasets**
EcoHAB, 6

all_toposorts, 2
arrange, 3

components, 3
cumulative_epochs, 4

diffuse, 4
dot, 5

EcoHAB, 6

marginise, 6

normalise, 7

parameters, 7

sample_toposort, 8

tally_from_matrix, 8
tally_log, 9
toporanga, 10

window_epochs, 11

zero_opposition, 12