

Package ‘timbr’

July 17, 2026

Type Package

Title Forest/Tree Data Frames

Version 0.3.0

Description Provides data frames for forest or tree data structures. You can create forest data structures from data frames and process them based on their hierarchies.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.1.0)

Imports cli, dplyr, lifecycle, pillar, purrr, rlang, tibble, tidygraph, tidyselect, vctrs (>= 0.5.2)

Suggests covr, testthat (>= 3.0.0)

Config/testthat/edition 3

URL <https://github.com/UchidaMizuki/timbr>,
<https://uchidamizuki.github.io/timbr/>

BugReports <https://github.com/UchidaMizuki/timbr/issues>

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Mizuki Uchida [aut, cre]

Maintainer Mizuki Uchida <uchidamizuki@vivaldi.net>

Repository CRAN

Date/Publication 2026-07-16 23:10:02 UTC

Contents

as_forest	2
children	2
climb	3
forest_by	3

2

children

leaves 4

map_forest 4

node_name 5

node_value 5

traverse 5

Index 7

as_forest	<i>Coerce to a forest</i>
-----------	---------------------------

Description

Coerce to a forest

Usage

as_forest(x, ...)

Arguments

- x An object.
- ... Unused, for extensibility.

Value

A forest.

children	<i>Children of the forest</i>
----------	-------------------------------

Description

Convert a forest into a forest consisting of its child nodes.

Usage

children(data, name = NULL)

Arguments

- data A forest.
- name NULL (default) or a scalar character specifying the node name of child nodes.

Value

A forest.

climb	<i>Climb a forest from parents to children</i>
-------	--

Description

Climb a forest from parents to children with one or more node names.

Usage

```
climb(.data, ..., .recurse = TRUE, .deep)
```

Arguments

.data	A forest.
...	A list of node names to climb the forest.
.recurse	Whether to search recursively by node names or not?
.deep	(Deprecated) Whether to search recursively by node names or not?

Value

A forest.

forest_by	<i>Constructs a forest by one or more variables</i>
-----------	---

Description

forest_by() constructs a forest by one or more variables.

Usage

```
forest_by(.data, ...)
```

Arguments

.data	A data frame.
...	Variables.

Value

A forest.

leaves	<i>Leaf nodes of a forest</i>
--------	-------------------------------

Description

Leaf nodes of a forest

Usage

```
leaves(data)
```

Arguments

data	A forest.
------	-----------

Value

A forest.

map_forest	<i>Apply a function hierarchically to a forest</i>
------------	--

Description**[Defunct]**

map_forest() has been removed. Please use [traverse\(\)](#) instead.

Usage

```
map_forest(.x, .f, ..., .climb = FALSE)
```

Arguments

.x	A forest
.f	A function, formula, or vector (not necessarily atomic).
...	Additional arguments passed on to the mapped function.
.climb	Climbing or descending?

Value

A forest.

node_name	<i>Get node names</i>
-----------	-----------------------

Description

Get node names

Usage

```
node_name()
```

Value

A character vector.

node_value	<i>Get node values</i>
------------	------------------------

Description

Get node values

Usage

```
node_value()
```

Value

A vector.

traverse	<i>Apply a function hierarchically to a forest</i>
----------	--

Description

Apply a function hierarchically to a forest in the climbing or descending direction.

Usage

```
traverse(.x, .f, ..., .climb = FALSE)
```

Arguments

<code>.x</code>	A forest
<code>.f</code>	A function, formula, or vector (not necessarily atomic).
<code>...</code>	Additional arguments passed on to the mapped function.
<code>.climb</code>	Climbing or descending?

Value

A forest.

Index

`as_forest`, [2](#)

`children`, [2](#)

`climb`, [3](#)

`forest_by`, [3](#)

`leaves`, [4](#)

`map_forest`, [4](#)

`node_name`, [5](#)

`node_value`, [5](#)

`traverse`, [5](#)

`traverse()`, [4](#)