# Package 'indexthis'

February 1, 2024

**Type** Package

**Title** Quick Indexation

**Version** 1.0.1

**Imports** Rcpp(>= 1.0.5)

**LinkingTo** Rcpp

**URL** https://github.com/lrberge/indexthis

**Depends** R(>= 3.5.0)

**Description** Quick indexation of any type of vector or of any combination of those. Indexation turns a vector into an integer vector going from 1 to the number of unique elements. Indexes are important building blocks for many algorithms. The method is described at <https://github.com/lrberge/indexthis/>.

**License** GPL-3

**RoxygenNote** 7.2.3

**Encoding** UTF-8

**NeedsCompilation** yes

**Author** Laurent Berge [aut, cre],
Sebastian Krantz [ctb],
Morgan Jacob [ctb]

**Maintainer** Laurent Berge <laurent.berge@u-bordeaux.fr>

**Repository** CRAN

**Date/Publication** 2024-02-01 17:40:02 UTC

## R topics documented:

---

to_index                          *Turns one or multiple vectors into an index (aka group id, aka key)*

---

## Description

Turns one or multiple vectors of the same length into an index, that is an integer vector of the same length ranging from 1 to the number of unique elements in the vectors. This is equivalent to creating a key.

## Usage

```
to_index(
  ...,
  list = NULL,
  sorted = FALSE,
  items = FALSE,
  items.simplify = TRUE,
  internal = FALSE
)
```

## Arguments

| | |
|---|---|
| `...` | The vectors to be turned into an index. Only works for atomic vectors. If multiple vectors are provided, they should all be of the same length. Notes that you can alternatively provide a list of vectors with the argument `list`. |
| `list` | An alternative to using `...` to pass the input vectors. If provided, it should be a list of atomic vectors, all of the same length. If this argument is provided, then `...` is ignored. |
| `sorted` | Logical, default is `FALSE`. By default the index order is based on the order of occurence. Values occurring before have lower index values. Use `sorted=TRUE` to have the index to be sorted based on the vector values. For example `c(7, 3, 7, -8)` will be turned into `c(1, 2, 1, 3)` if `sorted=FALSE` and into `c(3, 2, 3, 1)` is `sorted=TRUE`. |
| `items` | Logical, default is `FALSE`. Whether to return the input values the indexes refer to. If `TRUE`, a list of two elements, named `index` and `items`, is returned. The `items` object is a data.frame containing the values of the input vectors corresponding to the index. Note that if there is only one input vector and `items.simplify=TRUE` (default), then `items` is a vector instead of a data.frame. |
| `items.simplify` | Logical scalar, default is `TRUE`. Only used if the values from the input vectors are returned with `items=TRUE`. If there is only one input vector, the `items` is a vector if `items.simplify=TRUE`, and a data.frame otherwise. |
| `internal` | Logical, default is `FALSE`. If `TRUE`, some checks on the data are ignored. |

**Details**

The algorithm to create the indexes is based on a semi-hashing of the vectors in input. The hash table is of size 2 * n, with n the number of observations. Hence the hash of all values is partial in order to fit that range. That is to say a 32 bits hash is turned into a log2(2 * n) bits hash simply by shifting the bits. This in turn will necessarily lead to multiple collisions (ie different values leading to the same hash). This is why collisions are checked systematically, guaranteeing the validity of the resulting index.

Note that NA values are considered as valid and will not be returned as NA in the index. When indexing numeric vectors, there is no distinction between NA and NaN.

The algorithm is optimized for input vectors of type: i) numeric or integer (and equivalent data structures, like, e.g., dates), ii) logicals, iii) factors, and iv) character. The algorithm will be slow for types different from the ones previously mentioned, since a conversion to character will first be applied before indexing.

**Value**

By default, an integer vector is returned, of the same length as the inputs.

If you are interested in the values the indexes (i.e. the integer values) refer to, you can use the argument items = TRUE. In that case, a list of two elements, named index and items, is returned. The index is the integer vector representing the index, and the items is a data.frame containing the input values the index refers to.

Note that if items = TRUE and items.simplify = TRUE and there is only one vector in input, the items slot of the returned object will be equal to a vector.

**Author(s)**

Laurent Berge for this original implementation, Morgan Jacob (author of kit) and Sebastian Krantz (author of collapse) for the hashing idea.

**Examples**

```
x = c("u", "a", "a", "s", "u", "u")
y = c(  5,   5,   5,   3,   3,   7)

# By default, the index value is based on order of occurrence
to_index(x)
to_index(y)
to_index(x, y)

# Use the order of the input values with sorted=TRUE
to_index(x, sorted = TRUE)
to_index(y, sorted = TRUE)
to_index(x, y, sorted = TRUE)

# To get the values to which the index refer, use items=TRUE
to_index(x, items = TRUE)

# play around with the format of the output
```

```
to_index(x, items = TRUE, items.simplify = TRUE)   # => default
to_index(x, items = TRUE, items.simplify = FALSE)

# multiple items are always in a data.frame
to_index(x, y, items = TRUE)

# NAs are considered as valid
x_NA = c("u", NA, "a", "a", "s", "u", "u")
to_index(x_NA, items = TRUE)
to_index(x_NA, items = TRUE, sorted = TRUE)
```

# Index

to_index,