Package ‘ageutils’

September 9, 2024

Type Package

Title Collection of Functions for Working with Age Intervals

Version 0.0.5

Description Provides a collection of efficient functions for working with individual ages and corresponding intervals. These include functions for conversion from an age to an interval, aggregation of ages with associated counts in to intervals and the splitting of interval counts based on specified age distributions.

License GPL-2

Encoding UTF-8

RoxygenNote 7.3.2

Suggests dplyr, testthat (>= 3.0.0)

Depends R (>= 3.5.0)

LazyData true

URL `https://timtaylor.github.io/ageutils/`

BugReports `https://github.com/TimTaylor/ageutils/issues`

Config/build/compilation-database true

Imports cli, rlang, stats, tibble

Config/testthat/edition 3

NeedsCompilation yes

Author Tim Taylor [aut, cre, cph] (<`https://orcid.org/0000-0002-8587-7113`>), Edwin van Leeuwen [aut] (<`https://orcid.org/0000-0002-2383-5305`>)

Maintainer Tim Taylor <`tim.taylor@hiddenelephants.co.uk`>

Repository CRAN

Date/Publication 2024-09-09 21:50:02 UTC

Contents

`breaks_to_interval` *Convert breaks to an interval*

Description

`breaks_to_interval()` takes a specified set of breaks representing the left hand limits of a closed open interval, i.e [x, y), and returns the corresponding interval and upper bounds. The resulting intervals span from the minimum break through to a specified `max_upper`.

Usage

```
breaks_to_interval(breaks, max_upper = Inf)
```

Arguments

`breaks`	`[integerish]`. 1 or more non-negative cut points in increasing (strictly) order. These correspond to the left hand side of the desired intervals (e.g. the closed side of [x, y). Double values are coerced to integer prior to categorisation.
`max_upper`	`[numeric]` Represents the maximum upper bound splitting the data. Defaults to `Inf`.

Value

A tibble with an ordered factor column (`interval`), as well as columns corresponding to the explicit bounds (`lower_bound` and `upper_bound`). Note that even those these bounds are whole numbers they are returned as `numeric` to allow the maximum upper bound to be given as `Inf`.

Examples

```
breaks_to_interval(breaks = c(0L, 1L, 5L, 15L, 25L, 45L, 65L))
breaks_to_interval(
    breaks = c(0L, 1L, 5L, 15L, 25L, 45L, 65L),
    max_upper = 100L
)
```

cut_ages	*Cut integer age vectors*

Description

`cut_ages()` provides categorisation of ages based on specified breaks which represent the left-hand interval limits. The resulting intervals span from the minimum break through to a specified `max_upper` and will always be closed on the left and open on the right. Ages below the minimum break, or above `max_upper` will be returned as NA.

Usage

```
cut_ages(ages, breaks, max_upper = Inf)
```

Arguments

`ages`	`[numeric]`. Vector of age values. Double values are coerced to integer prior to categorisation / aggregation. Must not be NA.
`breaks`	`[integerish]`. 1 or more non-negative cut points in increasing (strictly) order. These correspond to the left hand side of the desired intervals (e.g. the closed side of [x, y). Double values are coerced to integer prior to categorisation.
`max_upper`	`[numeric]` Represents the maximum upper bound for the resulting intervals. Double values are rounded to the nearest (numeric) integer. Defaults to `Inf`.

Value

A data frame with an ordered factor column (`interval`), as well as columns corresponding to the explicit bounds (`lower_bound` and `upper_bound`).

Examples

```
cut_ages(ages = 0:9, breaks = c(0L, 3L, 5L, 10L))

cut_ages(ages = 0:9, breaks = c(0L, 5L))

# Note the following is comparable to a call to
# cut(ages, right = FALSE, breaks = c(breaks, Inf))
ages <- seq.int(from = 0, by = 10, length.out = 10)
breaks <- c(0, 1, 10, 30)
cut_ages(ages, breaks)
```

```
# values above max_upper treated as NA
cut_ages(ages = 0:10, breaks = c(0,5), max_upper = 7)
```

---

pop_dat *Aggregated population data*

Description

A dataset derived from the 2021 UK census containing population for different age categories across England and Wales.

Usage

```
pop_dat
```

Format

A data frame with 200 rows and 6 variables:

area_code Unique area identifier

area_name Unique area name

age_category Left-closed and right-open age interval

value count of individ

Source

https://github.com/TimTaylor/census_pop_2021

---

reaggregate_counts *Reaggregate age counts*

Description

`reaggregate_counts()` converts counts over one interval range to another with optional weighting by a known population.

Usage

```
reaggregate_counts(...)

## Default S3 method:
reaggregate_counts(
  bounds,
  counts,
  new_bounds,
  ...,
  population_bounds = NULL,
  population_weights = NULL
)
```

Arguments

`...`	Further arguments passed to or from other methods.
`bounds`	`[numeric]` The *current* boundaries in (strictly) increasing order. These correspond to the left hand side of the intervals (e.g. the closed side of [x, y). Double values are coerced to integer prior to categorisation.
`counts`	`[numeric]` Vector of counts corresponding to the intervals defined by `bounds`.
`new_bounds`	`[numeric]` The *desired* boundaries in (strictly) increasing order.
`population_bounds`	`[numeric]` Interval boundaries for a known population weighting given by the `population_weights` argument.
`population_weights`	`[numeric]` Population weightings corresponding to `population_bounds`. Used to weight the output across the desired intervals. If `NULL` (default), counts are divided proportional to the interval sizes.

Value

A data frame with 4 entries; `interval`, `lower_bound`, `upper_bound` and a corresponding `count`.

Examples

```
# Reaggregating some data obtained from the 2021 UK census
head(pop_dat)

# Each row of the data is for the same region so we can drop some columns
# `age_category` and `value` columns
dat <- subset(pop_dat, select = c(age_category, value))
```

```
# Add the lower bounds to the data
dat <- transform(
    dat,
    lower_bound = as.integer(sub("\\[([0-9]+), .+)", "\\1", age_category))
)

# Now recategorise to the desired age intervals
with(
    dat,
    reaggregate_counts(
        bounds = lower_bound,
        counts = value,
        new_bounds = c(0L, 1L, 5L, 15L, 25L, 45L, 65L)
    )
)
```

`reaggregate_rates` *Reaggregate age rates*

Description

`reaggregate_rates()` converts rates over one interval range to another with optional weighting by a known population.

Usage

```
reaggregate_rates(...)

## Default S3 method:
reaggregate_rates(
  bounds,
  rates,
  new_bounds,
  ...,
  population_bounds = NULL,
  population_weights = NULL
)
```

Arguments

`...`	Further arguments passed to or from other methods.
`bounds`	`[numeric]` The *current* boundaries in (strictly) increasing order. These correspond to the left hand side of the intervals (e.g. the closed side of [x, y). Double values are coerced to integer prior to categorisation.

`rates`	`[numeric]` Vector of rates corresponding to the intervals defined by `bounds`.
`new_bounds`	`[numeric]` The *desired* boundaries in (strictly) increasing order.
`population_bounds`	`[numeric]` Interval boundaries for a known population weighting given by the `population_weights` argument.
`population_weights`	`[numeric]` Population weightings corresponding to `population_bounds`. Used to weight the output across the desired intervals. If `NULL` (default) rates are divided proportional to the interval sizes.

Value

A data frame with 4 entries; `interval`, `lower_bound`, `upper_bound` and a corresponding `rate`.

Examples

```
reaggregate_rates(
    bounds = c(0, 5, 10),
    rates = c(0.1, 0.2 ,0.3),
    new_bounds = c(0, 2, 7, 10),
    population_bounds = c(0, 2, 5, 7, 10),
    population_weights = c(100, 200, 50, 150, 100)
)
```

Index