Package 'ageutils'

June 26, 2024

Type Package

Title Collection of Functions for Working with Age Intervals

Version 0.0.4

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 7.3.1

Suggests knitr, markdown, tinytest

Depends R (>= 3.5.0)

LazyData true

VignetteBuilder knitr

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

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

Config/runiverse/noindex true

NeedsCompilation yes

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

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

Repository CRAN

Date/Publication 2024-06-26 21:20:06 UTC

Contents

`aggregate_age_counts` *Aggregate counts across ages*

Description

`aggregate_age_counts()` provides aggregation of counts across ages (in years). It is similar to a `cut()` and `tapply()` pattern but optimised for speed over flexibility. It 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 the maximum age. Missing values are grouped as NA.

Usage

```
aggregate_age_counts(counts, ages = seq_along(counts) - 1L, breaks)
```

Arguments

`counts`	`[numeric]`.
	Vector of counts to be aggregated.
`ages`	`[numeric]`.
	Vector of age in years.
	Double values are coerced to integer prior to categorisation / aggregation.
	For `aggregate_age_counts()`, these must corresponding to the `counts` entry and will defaults to 0:(N-1) where `N` is the number of counts present.
	No (non-missing) age can be less than the minimum break.
`breaks`	`[numeric]`.
	1 or more 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.

Value

A data frame with 4 entries; `interval`, `lower_bound`, `upper_bound` and an associated `count`.

Examples

```
# default ages generated if only counts provided (here ages will be 0:64)
aggregate_age_counts(counts = 1:65, breaks = c(0L, 1L, 5L, 15L, 25L, 45L, 65L))

# NA ages are handled with their own grouping
ages <- 1:65
ages[1:44] <- NA
aggregate_age_counts(
    counts = 1:65,
    ages = ages,
    breaks = c(0L, 1L, 5L, 15L, 25L, 45L, 65L)
)
```

---

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 data frame with an ordered factor column (`interval`), as well as columns corresponding to the explicit bounds (`lower_bound` and `upper_bound`). Note these bounds are returned as `<numeric>` to allow the maximum upper bound to be `Inf`.

Examples

```
brks <- c(0L, 1L, 5L, 15L, 25L, 45L, 65L)
breaks_to_interval(breaks = brks)
breaks_to_interval(breaks = brks, 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_interval_counts *Reaggregate age intervals*

Description

reaggregate_interval_counts() converts counts over one interval range to another. It first splits counts of a given age interval in to counts for individual years based on a given weighting. These are then aggregated to the desired breaks. Functionally this is equivalent to, but more efficient than, a call to split_interval_counts() followed by aggregate_age_counts().

Usage

```
reaggregate_interval_counts(
  lower_bounds,
  upper_bounds,
  counts,
  breaks,
  max_upper = 100L,
  weights = NULL
)
```

Arguments

lower_bounds, upper_bounds
: [integerish].

  A pair of vectors representing the bounds of the intervals.

  lower_bounds must be strictly less than upper_bounds and greater than or equal to zero.

  Missing (NA) bounds are not permitted.

  Double vectors will be coerced to integer.

counts
: [numeric].

  Vector of counts to be aggregated.

breaks
: [numeric].

  1 or more 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
: [integerish]

  Represents the maximum upper bounds permitted upon splitting the data.

  Any upper bound greater than this will be replaced with this value prior to splitting.

  Double vectors will be coerced to integer.

weights [numeric]

Population weightings to apply for individual years.

If NULL (default) counts will be split evenly based on interval size.

If specified, must be of length max_upper and represent weights in the range 0:(max_upper - 1).

Value

A data frame with 4 entries; interval, lower_bound, upper_bound and an associated count.

Examples

```
reaggregate_interval_counts(
    lower_bounds = c(0, 5, 10),
    upper_bounds = c(5, 10, 20),
    counts = c(5, 10, 30),
    breaks = c(0L, 1L, 5L, 15L, 25L, 45L, 65L)
)
```

---

reaggregate_interval_rates

Reaggregate rates across intervals

Description

reaggregate_interval_rates() enables the reweighting of interval rates in to different intervals ranges. It first replicates the rates of a given age interval into the individual years of said interval. These are then aggregated allowing for a user specified weight vector.

Usage

```
reaggregate_interval_rates(
  lower_bounds,
  upper_bounds = NULL,
  rates,
  breaks,
  weights = NULL
)
```

Arguments

lower_bounds, upper_bounds

[integerish].

A pair of vectors representing the bounds of the current intervals.

If upper_bounds is NULL, it will be automatically set to c(lower_bounds[-1L], Inf).

	lower_bounds must be strictly less than upper_bounds and greater than or equal to zero.
	Missing (NA) bounds are not permitted.
	Double vectors will be coerced to integer.
rates	[numeric].
	Vector of counts to be averaged.
breaks	[numeric].
	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.
weights	[numeric]
	Population weightings to apply for individual years.
	If NULL (default) weights will be allocated proportional to the interval size.
	If specified, must be of length most 2000 and represent weights in the range 0:1999.
	weights of length less than 2000 will be padded with 0.

Value

A data frame with 4 entries; interval, lower_bound, upper_bound and an associated count.

Examples

```
reaggregate_interval_rates(
  lower_bounds = c(0, 5, 13),
  upper_bounds= c(5, 15, 100),
  rates = c(1, 0.1, 0.01),
  breaks = c(0, 1, 9, 15),
  weights = round(runif(70, 10, 30))
)

reaggregate_interval_rates(
  lower_bounds = c(0, 5, 13),
  rates = c(1, 0.1, 0.01),
  breaks = c(0, 1, 9, 15),
  weights = round(runif(70, 10, 30))
)
```

---

split_interval_counts *Split interval counts*

Description

`split_interval_counts()` splits counts of a given age interval in to counts for individual years based on a given weighting. Age intervals are specified by their lower (closed) and upper (open) bounds, i.e. intervals of the form [lower, upper).

Usage

```
split_interval_counts(
  lower_bounds,
  upper_bounds,
  counts,
  max_upper = 100L,
  weights = NULL
)
```

Arguments

`lower_bounds`, `upper_bounds`

`[integerish]`.

A pair of vectors representing the bounds of the intervals.

`lower_bounds` must be strictly less than `upper_bounds` and greater than or equal to zero.

Missing (NA) bounds are not permitted.

Double vectors will be coerced to integer.

`counts` `[numeric]`.

Vector of counts to be aggregated.

`max_upper` `[integerish]`

Represents the maximum upper bounds permitted upon splitting the data.

Any upper bound greater than this will be replaced with this value prior to splitting.

Double vectors will be coerced to integer.

`weights` `[numeric]`

Population weightings to apply for individual years.

If `NULL` (default) counts will be split evenly based on interval size.

If specified, must be of length `max_upper` and represent weights in the range 0:(max_upper - 1).

Value

A data frame with entries `age` (in years) and `count`.

Examples

```
split_interval_counts(
    lower_bounds = c(0, 5, 10),
    upper_bounds = c(5, 10, 20),
    counts = c(5, 10, 30)
```

```
)

split_interval_counts(
    lower_bounds = c(0, 5, 10),
    upper_bounds = c(5, 10, Inf),
    counts = c(5, 10, 30),
    max_upper = 15
)

split_interval_counts(
    lower_bounds = c(0, 5),
    upper_bounds = c(5, 10),
    counts = c(5, 10),
    max_upper =10,
    weights = 1:10
)
```

Index