

# Package ‘ageutils’

May 7, 2026

**Type** Package

**Title** Collection of Functions for Working with Age Intervals

**Version** 0.1.2

**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** 7.3.3

**Imports** rlang, stats, tibble, vctrs

**Suggests** dplyr, litedown, 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/testthat/edition** 3

**Config/testthat/load-all** list(export\_all = FALSE, helpers = TRUE)

**VignetteBuilder** litedown

**NeedsCompilation** no

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

**Maintainer** Tim Taylor <[tim.taylor@hiddenelephants.co.uk](mailto:tim.taylor@hiddenelephants.co.uk)>

**Repository** CRAN

**Date/Publication** 2026-02-27 12:30:02 UTC

## Contents

breaks_to_interval . . . . .	2
cut_ages . . . . .	3
pop_dat . . . . .	4
reaggregate_counts . . . . .	4
reaggregate_rates . . . . .	6

<b>Index</b>	<b>8</b>
--------------	----------

---

breaks_to_interval	<i>Convert breaks to an interval</i>
--------------------	--------------------------------------

---

### 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 and upper). 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(0, 1, 5, 15, 25, 45, 65))
breaks_to_interval(
  breaks = c(0, 1, 5, 15, 25, 45, 65),
  max_upper = 100
)
```

---

`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 above `max_upper` will be returned as NA.

### Usage

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

### Arguments

<code>ages</code>	[numeric]. Vector of age values. Double values are coerced to integer prior to categorisation / aggregation. Must not be NA.
<code>breaks</code>	[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).
<code>max_upper</code>	[numeric] Represents the maximum upper bound for the resulting intervals. Double values are rounded up to the nearest (numeric) integer. Defaults to Inf.

### Value

A [tibble](#) with an ordered factor column (`interval`), as well as columns corresponding to the explicit bounds (`lower` and `upper`). Internally both bound columns are stored as double but it can be taken as part of the function API that `lower` is coercible to integer without any coercion to `NA_integer_`. Similarly all values of `upper` apart from those corresponding to `max_upper` can be assumed coercible to integer (`max_upper` may or may not depending on the given argument).

### Examples

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

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

# 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](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(
  bounds,
  counts,
  new_bounds,
  ...,
  population_bounds = NULL,
  population_weights = NULL
)
```

**Arguments**

bounds	[numeric]	The <i>current</i> 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 <i>desired</i> boundaries in (strictly) increasing order.
...		Further arguments passed to or from other methods.
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 [tibble](#) with 4 entries; interval, lower, upper 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(0, 1, 5, 15, 25, 45, 65)
  )
)

```

---

reaggregate_rates	<i>Reaggregate age rates</i>
-------------------	------------------------------

---

## Description

reaggregate\_rates() converts rates over one interval range to another with optional weighting by a known population.

## Usage

```

reaggregate_rates(
  bounds,
  rates,
  new_bounds,
  ...,
  population_bounds = NULL,
  population_weights = NULL
)

```

## Arguments

bounds	<p>[numeric]</p> <p>The <i>current</i> boundaries in (strictly) increasing order. These correspond to the left hand side of the intervals (e.g. the closed side of [x, y]).</p> <p>Double values are coerced to integer prior to categorisation.</p>
rates	<p>[numeric]</p> <p>Vector of rates corresponding to the intervals defined by bounds.</p>
new_bounds	<p>[numeric]</p> <p>The <i>desired</i> boundaries in (strictly) increasing order.</p>
...	<p>Further arguments passed to or from other methods.</p>

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`, `upper` 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

\* **datasets**

pop\_dat, [4](#)

breaks\_to\_interval, [2](#)

cut\_ages, [3](#)

pop\_dat, [4](#)

reaggregate\_counts, [4](#)

reaggregate\_rates, [6](#)

tibble, [2](#), [3](#), [5](#)