Package ‘poputils’

July 23, 2025

Type Package

Title Demographic Analysis and Data Manipulation

Version 0.4.2

Description Perform tasks commonly encountered when
preparing and analysing demographic data.
Some functions are intended for end users, and
others for developers. Includes functions for
working with life tables.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Depends R (>= 4.3.0)

LinkingTo cpp11

Imports cli, rlang, rvec, tibble, tidyselect, utils, vctrs

Suggests bookdown, covr, dplyr, ggplot2, knitr, rmarkdown, testthat
(>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

URL https://bayesiandemography.github.io/poputils/,
https://github.com/bayesiandemography/poputils

BugReports https://github.com/bayesiandemography/poputils/issues

NeedsCompilation yes

Author John Bryant [aut, cre],
Bayesian Demography Limited [cph]

Maintainer John Bryant <john@bayesiandemography.com>

Repository CRAN

Date/Publication 2025-07-12 04:40:02 UTC

Contents

`age_group_type` *Infer Age Label Type*

Description

Determine whether a set of age labels refer to one-year, five-year, or life-table age groups.

Usage

```
age_group_type(x)
```

Arguments

`x`	A vector of age labels

Details

The valid types of age labels are:

- "single". One-year age groups, eg "0" or "55", and possibly an open age group, eg "90+".
- "five". Five-year age groups, eg "0-4" or "55-59", and possibly an open age group, eg "100+".
- "lt". Life table age groups, eg "0", "1-4", "5-9", "55-59", or "80+".

If x does not fit any of these descriptions, then age_group_type() throws an error.

If x could belong to more than one type, then age_group_type() prefers "single" to "five" and "lt", and prefers "five" to "lt".

Value

"single", "five", or "lt".

Examples

```
age_group_type(c("5-9", "0-4", "100+"))
age_group_type(c("2", "5", "1"))
age_group_type(c("0", "1-4"))

## could be any "single" or "lt"
age_group_type("0")

## could be "five" or "lt"
age_group_type("80-84")
```

---

age_labels *Create Age Labels*

Description

Create labels for age groups. The labels depend on the type argument:

- "single". One-year age groups, eg "0" or "55", and possibly an open age group, eg "90+".
- "five". Five-year age groups, eg "0-4" or "55-59", and possibly an open age group, eg "100+".
- "lt". Life table age groups, eg "0", "1-4", "5-9", "55-59", or "80+".

Usage

```
age_labels(type, min = 0, max = 100, open = NULL)
```

Arguments

type	Type of age group labels: "single", "five", or "lt".
min	Minimum age. Defaults to 0.
max	Maximum age for closed age groups. Defaults to 100.
open	Whether the last age group is "open", ie has no upper limit.

Details

The first age group starts at the age specified by min. If open is TRUE, then the final age group starts at the age specified by max. Otherwise, the final age group ends at the age specified by max.

open defaults to TRUE when min equals zero, and to FALSE otherwise.

Value

A character vector.

See Also

reformat_age()

Examples

```
age_labels(type = "single", min = 15, max = 40)
age_labels(type = "five")
age_labels(type = "lt", max = 80)
```

---

age_lower *Lower Limits, Midpoints, and Upper Limits of Age Groups*

Description

Given a vector x of age group labels, return a numeric vector.

- age_lower() returns the lower limits of each age group,
- age_mid() returns the midpoints, and
- age_upper() returns the upper limits.

Vector x must describe 1-year, 5-year or life-table age groups: see age_labels() for examples. x can format these age groups in any way understood by reformat_age().

Usage

```
age_lower(x)

age_mid(x)

age_upper(x)
```

Arguments

x A vector of age group labels.

Details

These functions can make age groups easier to work with. Lower and upper limits can be used for selecting on age. Replacing age group with midpoints can improve graphs.

Value

A numeric vector, the same length as x.

See Also

reformat_age() age_labels()

Examples

```
x <- c("15-19", "5-9", "50+")
age_lower(x)
age_mid(x)
age_upper(x)

## non-standard formats are OK
age_lower(c("infants", "100 and over"))

df <- data.frame(age = c("1-4", "10-14", "5-9", "0"),
                 rate = c(0.023, 0.015, 0.007, 0.068))
df
subset(df, age_lower(age) >= 5)
```

---

check_age *Validity Checks for Age Labels*

Description

Check that age labels can be parsed and, optionally, whether the labels are complete, unique, start at zero, and end with an open age group.

Usage

```
check_age(
  x,
  complete = FALSE,
  unique = FALSE,
  zero = FALSE,
  open = FALSE,
  closed = FALSE
)
```

Arguments

`x`	A vector of age labels.
`complete`	If `TRUE`, test whether `x` has gaps.
`unique`	If `TRUE`, test whether `x` has duplicates.
`zero`	If `TRUE`, test whether youngest age group in `x` starts at 0.
`open`	If `TRUE`, test whether oldest age group in `x` is open.
`closed`	If `TRUE`, test whether oldest age group in `x` is closed.

Details

By default, `check_age()` only tests whether a set of labels can be parsed as single-year, five-year, or life table age groups. (See `age_group_type()` for more on the three types of age group.) However, it can also apply the following tests:

- `complete`. Whether `x` includes all intermediate age groups, with no gaps. For instance, the labels `c("10-14", "15-19", "5-9")` are complete, while the labels`c("15-19", "5-9")` are not (because they are missing `"10-14"`.)
- `unique`. Whether `x` has duplicated labels.
- `zero`. Whether the youngest age group in `x` starts at age 0, ie whether it includes `"0"` or `"0-4"`.
- `open`. Whether the oldest age group in `x` is "open", with no upper limit, eg `"100+"` or `"65+"`.
- `closed`. Whether the oldest age group in `x` is "closed", with an upper limit, eg `"100-104+"` or `"65"`.

Value

`TRUE`, invisibly, or raises an error if a test fails.

See Also

- `reformat_age()` to convert age labels to the format used by **poputils**.

Examples

```
try(
  check_age(c("10-14", "0-4", "15+"),
            complete = TRUE)
)

try(
  check_age(c("10-14", "5-9", "0-4", "5-9", "15+"),
            unique = TRUE)
)

try(
  check_age(c("10-14", "5-9", "15+"),
            zero = TRUE)
)
```

```
try(
  check_age(c("10-14", "0-4", "5-9"),
            open = TRUE)
)

try(
  check_age(c("10+", "0-4", "5-9"),
            closed = TRUE)
)
```

`check_equal_length` *Check that Arguments have Same Length*

Description

Check that `x` and `y` have the same length.

Usage

```
check_equal_length(x, y, nm_x, nm_y)
```

Arguments

`x, y`	Arguments to compare
`nm_x, nm_y`	Names to use in error message

Value

'TRUE', invisibly.

Examples

```
x <- 1:3
y <- 3:1
check_equal_length(x = x,
                   y = y,
                   nm_x = "x",
                   nm_y = "y")
```

check_n *Check Whole Number*

Description

Check that n is finite, non-NA scalar that is an integer or integerish (ie is equal to round(n)), and optionally within a specified range and divisible by a specified number.

Usage

```
check_n(n, nm_n, min, max, divisible_by)
```

Arguments

n	A whole number
nm_n	Name for 'n' to be used in error messages
min	Minimum value 'n' can take. Can be NULL.
max	Maximum values 'n' can take. Can be NULL.
divisible_by	'n' must be divisible by this. Can be NULL.

Value

If all tests pass, check_n() returns TRUE invisibly. Otherwise it throws an error.

Examples

```
check_n(10, nm_n = "count", min = 0, max = NULL, divisible_by = 1)
check_n(10, nm_n = "count", min = NULL, max = NULL, divisible_by = NULL)
check_n(10, nm_n = "n", min = 5, max = 10, divisible_by = 2)
```

check_no_overlap_colnums *Check that Colnum Vectors do not Overlap*

Description

Given a named list of colnum vectors, like those produced by tidyselect::eval_select(), throw an error if there is an overlap.

Usage

```
check_no_overlap_colnums(x)
```

Arguments

x	A named list of integer vectors.

Value

`TRUE`, invisibly

See Also

`tidyselect::eval_select()`

Examples

```
x <- list(arg1 = c(age = 1L),
          arg2 = c(gender = 4L, region = 5L))
check_no_overlap_colnums(x)
```

---

combine_age *Aggregate Age Group Labels*

Description

Convert age group labels to a less detailed classification. The three classifications recognized by `combine_age()` are `"single"`, `"five"`, and `"lt"`, as defined on `age_labels()`. The following conversions are permitted:

- `"single"` —> `"lt"`
- `"single"` —> `"five"`
- `"lt"` —> `"five"`

Usage

```
combine_age(x, to = c("five", "lt"))
```

Arguments

`x`	A vector of age labels
`to`	Type of age classification to convert to: `"five"` or `"lt"`. Defaults to `"five"`.

Value

If `x` is a factor, then `combine_age()` returns a factor; otherwise it returns a character vector.

See Also

- `age_labels()` to create age group labels
- `reformat_age()` to convert existing age group labels to a standard format
- `set_age_open()` to set the lower limit of the open age group

Examples

```
x <- c("0", "5", "3", "12")
combine_age(x)
combine_age(x, to = "lt")
```

`ex_to_lifetab_brass` *Derive Life Tables that Match Life Expectancies, using a Brass Logit Model*

Description

Turn life expectancies at birth into full life tables, using the Brass logit model. The method is simple and is designed for simulations or for settings with little or no data on age-specific mortality rates. In settings where data on age-specific mortality is available, other methods might be more appropriate.

Usage

```
ex_to_lifetab_brass(
  target,
  standard,
  infant = c("constant", "linear", "CD", "AK"),
  child = c("constant", "linear", "CD"),
  closed = c("constant", "linear"),
  open = "constant",
  radix = 1e+05,
  suffix = NULL
)
```

Arguments

`target` A data frame containing a variable called `"ex"`, and possibly others. See Details.

`standard` A data frame containing variables called `age` and `lx`, and possibly others. See details.

`infant, child, closed, open` Methods used to calculate life expectancy. See `lifetab()` for details.

`radix` Initial population for the `lx` column in the derived life table(s). Default is `100000`.

`suffix` Optional suffix added to life table columns.

Value

A data frame containing one or more life tables.

Method

The method implemented by `ex_to_lifetab_brass()` is based on the observation that, if populations A and B are demographically similar, then, in many cases,

$$\text{logit}(l_x^{\text{B}}) \approx \alpha + \beta \text{logit}(l_x^{\text{A}})$$

where l_x is the "survivorship probability" quantity from a life table. When populations are similar, $beta$ is often close to 1.

Given (i) target life expectancy, (ii) a set of l_x^{A}), (referred to as a "standard"), and (iii) a value for β, `ex_to_lifetab_brass()` finds a value for α that yields a set of l_x^{B}) with the required life expectancy.

`target` argument

`target` is a data frame specifying life expectancies for each population being modelled, and, possibly, inputs to the calculations, and index variables. Values in `target` are not age-specific.

- A variable called `"ex"`, with life expectancy at birth must be included in `target`.
- A variable called `"beta"` with values for `beta` can be included in `target`. This variable can be an rvec. If no `"beta"` variable is included in `target`, then `ex_to_lifetab_brass()` assumes that $beta \equiv 1$.
- A variable called `"sex"`. If the `infant` argument to `ex_to_lifetab_brass()` is is `"CD"` or `"AK"`, or if the `child` argument is `"CD"`, `target` must include a `"sex" variable, and the labels for this variable` variable is optional, and there is no restriction on labels.
- Other variables used to distinguish between life expectancies, such as time, region, or model variant.

`standard` argument

`standard` is a data frame specifying the l_x to be used with each life expectancy in `ex`, and, optionally, values the average age person-years lived by people who die in each group, ${}_n a_x$. Values in `standard` are age-specific.

- A variable called `"age"`, with labels that can be parsed by `reformat_age()`.
- A variable called `"lx"`. Internally each set of l_x is are standardized so that the value for age 0 equals 1. Within each set, values must be non-increasing. Cannot be an rvec.
- Additional variables used to match rows in `standard` to rows in `target`.

Internally, `standard` is merged with `target` using a left join from `target`, on any variables that `target` and `standard` have in common.

References

Brass W, Coale AJ. 1968. “Methods of analysis and estimation,” in Brass, W, Coale AJ, Demeny P, Heisel DF, et al. (eds). The Demography of Tropical Africa. Princeton NJ: Princeton University Press, pp. 88–139.

Moultrie TA, Timæus IM. 2013. Introduction to Model Life Tables. In Moultrie T, Dorrington R, Hill A, Hill K, Timæus I, Zaba B. (eds). Tools for Demographic Estimation. Paris: International Union for the Scientific Study of Population. online version.

See Also

- logit(), invlogit() Logit function
- lifeexp() Calculate life expectancy from detailed inputs

Examples

```
## create new life tables based on level-1
## 'West' model life tables, but with lower
## life expectancy

library(dplyr, warn.conflicts = FALSE)

target <- data.frame(sex = c("Female", "Male"),
                     ex = c(17.5, 15.6))

standard <- west_lifetab |>
    filter(level == 1) |>
    select(sex, age, lx)

ex_to_lifetab_brass(target = target,
                    standard = standard,
                    infant = "CD",
                    child = "CD")
```

---

find_label_female   *Identify Sex or Gender Labels Referring to Females*

Description

Given labels for sex or gender, try to infer which (if any) refer to females. If no elements look like a label for females, or if two or more elements do, then return NULL.

Usage

```
find_label_female(nms)
```

Arguments

nms	A character vector

Value

An element of nms or NULL.

See Also

find_label_male(), find_var_sexgender()

Examples

```
find_label_female(c("Female", "Male")) ## one valid
find_label_female(c("0-4", "5-9"))     ## none valid
find_label_female(c("F", "Fem"))       ## two valid
```

`find_label_male` *Identify Sex or Gender Labels Referring to Males*

Description

Given labels for sex or gender, try to infer which (if any) refer to males. If no elements look like a label for males, or if two or more elements do, then return `NULL`.

Usage

```
find_label_male(nms)
```

Arguments

`nms`	A character vector

Value

An element of `nms` or `NULL`.

See Also

`find_label_female()`, `find_var_sexgender()`

Examples

```
find_label_male(c("Female", "Male")) ## one valid
find_label_male(c("0-4", "5-9"))     ## none valid
find_label_male(c("male", "m"))      ## two valid
```

find_var_age — *Identify an Age Variable*

Description

Find the element of nms that looks like an age variable. If no elements look like an age variable, or if two or more elements do, then return NULL.

Usage

```
find_var_age(nms)
```

Arguments

nms	A character vector

Value

An element of nms, or NULL.

See Also

find_var_time(), find_var_sexgender()

Examples

```
find_var_age(c("Sex", "Year", "AgeGroup", NA)) ## one valid
find_var_age(c("Sex", "Year"))                 ## none valid
find_var_age(c("age", "age.years"))            ## two valid
```

find_var_sexgender — *Identify a Sex or Gender Variable*

Description

Find the element of nms that looks like a sex or gender variable. If no elements look like a sex or gender variable, or if two or more elements do, then return NULL.

Usage

```
find_var_sexgender(nms)
```

Arguments

nms	A character vector

Value

An element of nms, or NULL.

See Also

find_var_age(), find_var_time(), find_label_female(), find_label_male()

Examples

```
find_var_sexgender(c("Sex", "Year", "AgeGroup", NA)) ## one valid
find_var_sexgender(c("Age", "Region"))               ## none valid
find_var_sexgender(c("sexgender", "sexes"))          ## two valid
```

---

find_var_time — *Identify a Time Variable*

Description

Find the element of nms that looks like an time variable. If no elements look like a time variable, or if two or more elements do, then return NULL.

Usage

```
find_var_time(nms)
```

Arguments

nms	A character vector

Value

An element of nms, or NULL.

See Also

find_var_age(), find_var_sexgender()

Examples

```
find_var_time(c("Sex", "Year", "AgeGroup", NA)) ## one valid
find_var_time(c("Sex", "Region"))               ## none valid
find_var_time(c("time", "year"))                ## two valid
```

groups_colnums — *Get a named vector of column indices for the grouping variables in a grouped data frame*

Description

Constructed a named vector of indices equivalent to the vectors produced by tidyselect::eval_select, but for the grouping variables in an object of class "grouped_df".

Usage

```
groups_colnums(data)
```

Arguments

`data`	A data frame.

Details

If `data` is not grouped, then `groups_colnums` returns a zero-length vector.

Value

A named integer vector.

Examples

```
library(dplyr)
df <- data.frame(x = 1:4,
                 g = c(1, 1, 2, 2))
groups_colnums(df)
df <- group_by(df, g)
groups_colnums(df)
```

iran_fertility — *Age-Specific Fertility Rates in Iran*

Description

Estimates of age-specific fertility rates, (births per 1000 person-years lived) for rural and urban areas, in Iran, 1986-2000. Calculated by Mohammad Jalal Abbasi-Shavazi and Peter McDonald from data from the 2000 Iran Demographic and Health Survey.

Usage

```
iran_fertility
```

Format

A tibble with 2010 rows and the following columns:

- time Calendar year
- age Five-year age group from `"15-19"` to `"45-49"`
- area `"Rural" or` "Urban"‘
- rate Age-specific fertility rate

Source

Tables 4.1 and 4.2 of Abbasi-Shavazi, M J, McDonald, P (2005). *National and provincial level fertility trends in Iran, 1972–2006.* Australian National University. Working Papers in Demography no. 94.

`lifetab` *Calculate Life Tables or Life Expectancies*

Description

Calculate life table quantities. Function `lifetab()` returns an entire life table. Function `lifeexp()` returns life expectancy at birth. The inputs can be mortality rates (`mx`) or probabilities of dying (`qx`), though not both.

Usage

```
lifetab(
  data,
  mx = NULL,
  qx = NULL,
  age = age,
  sex = NULL,
  ax = NULL,
  by = NULL,
  infant = c("constant", "linear", "CD", "AK"),
  child = c("constant", "linear", "CD"),
  closed = c("constant", "linear"),
  open = "constant",
  radix = 1e+05,
  suffix = NULL,
  n_core = 1
)

lifeexp(
  data,
  mx = NULL,
  qx = NULL,
```

```
  at = 0,
  age = age,
  sex = NULL,
  ax = NULL,
  by = NULL,
  infant = c("constant", "linear", "CD", "AK"),
  child = c("constant", "linear", "CD"),
  closed = c("constant", "linear"),
  open = "constant",
  suffix = NULL,
  n_core = 1
)
```

Arguments

data	Data frame with mortality data.
mx	<tidyselect> Mortality rates, expressed as deaths per person-year lived. Possibly an rvec.
qx	<tidyselect> Probability of dying within age interval. An alternative to mx. Possibly an rvec.
age	<tidyselect> Age group labels. The labels must be interpretable by functions such as reformat_age() and age_group_type(). The first age group must start at age 0, and the last age group must be "open", with no upper limit.
sex	<tidyselect> Biological sex, with labels that can be interpreted by reformat_sex(). Needed only when infant is "CD" or "AK", or child is "CD".
ax	<tidyselect> Average age at death within age group. Optional. See Details.
by	<tidyselect> Separate life tables, or life expectancies, calculated for each combination the by variables. If a sex variable was specified, then that variable is automatically included among the by variables. If data is a grouped data frame, then the grouping variables take precedence over by.
infant	Method used to calculate life table values in age group "0". Ignored if age does not include age group "0". Default is "constant".
child	Method used to calculate life table values in age group "1-4". Ignored if age does not include age group "0". Default is "constant".
closed	Method used to calculate life table values in closed age intervals other than "0" and "1-4" (ie intervals such as "10-14" or "12"). Default is "constant".
open	Method used to calculate life table values in the final, open age group (eg "80+" or "110+"). Currently the only option is ‘"constant".
radix	Initial population for the lx column. Default is 100000.
suffix	Optional suffix added to new columns in result.
n_core	Number of cores to use for parallel processing. If n_core is 1 (the default), no parallel processing is done.
at	Age at which life expectancy is calculated (lifeexp() only). Default is 0‘. Can be a vector with length > 1.

Value

A tibble.

Definitions of life table quantities

- mx Deaths per person-year lived.
- qx Probability of surviving from the start of age group 'x' to the end.
- lx Number of people alive at the start of age group x.
- dx Number of deaths in age group x
- Lx Expected number of person years lived in age group x.
- ex Life expectancy, calculated at the start of age group x.

Mortality rates mx are sometimes expressed as deaths per 1000 person-years lived, or per 100,000 person-years lived. `lifetab()` and `lifeexp()` assumed that they are expressed as deaths per person-year lived.

Calculation methods

`lifetab()` and `lifeexp()` implement several methods for calculating life table quantities from mortality rates. Each method makes different assumptions about the way that mortality rates vary within age intervals:

- "constant" Mortality rates are constant within each interval.
- "linear". Life table quantity lx is a straight line within each interval. Equivalently, deaths are distributed uniformly within each interval.
- "CD". Used only with age groups "0" and "1-4". Mortality rates decline over the age interval, with the slope depending on the absolute level of infant mortality. The formulas were developed by Coale and Demeny (1983), and used in Preston et al (2001).
- "AK". Used only with age group "0". Mortality rates decline over the age interval, with the slope depending on the absolute level of infant mortality. The formulas were formulas developed by Andreev and Kingkade (2015), and are used in the Human Mortality Database methods protocol.

For a detailed description of the methods, see the vignette for **poputils**.

ax

ax is the average number of years lived in an age interval by people who die in that interval. Demographers sometimes refer to it as the 'separation factor'. If a non-NA value of ax is supplied for an age group, then the results for that age group are based on the formula

$$m_x = d_x / (n_x l_x + a_x d_x)$$

,

(where n_x is the width of the age interval), over-riding any methods specified via the infant, child, closed and open arguments.

Open age group when inputs are qx

The probability of dying, `qx`, is always 1 in the final (open) age group. `qx` therefore provides no direct information on mortality conditions within the final age group. `lifetab()` and `lifeexp()` use conditions in the second-to-final age group as a proxy for conditions in the final age group. When `open` is `"constant"` (which is currently the only option), and no value for `ax` in the final age group is provided, `lifetab()` and `lifeexp()` assume that $m_A = m_{A-1}$, and set $L_A = l_A / m_A$.

In practice, mortality is likely to be higher in the final age group than in the second-to-final age group, so the default procedure is likely to lead to inaccuracies. When the size of the final age group is very small, these inaccuracies will be inconsequential. But in other cases, it may be necessary to supply an explicit value for `ax` for the final age group, or to use `mx` rather than `qx` as inputs.

Using rvecs to represent uncertainty

An rvec is a 'random vector', holding multiple draws from a distribution. Using an rvec for the `mx` argument to `lifetab()` or `lifeexp()` is a way of representing uncertainty. This uncertainty is propagated through to the life table values, which will also be rvecs.

Parallel processing

Calculations can be slow when working with rvecs and many combinations of 'by' variables. In these cases, setting `n_core` to a number greater than 1, which triggers parallel processing, may help.

References

- Preston SH, Heuveline P, and Guillot M. 2001. *Demography: Measuring and Modeling Population Processes* Oxford: Blackwell.
- Coale AJ, Demeny P, and Vaughn B. 1983. *Regional model life tables and stable populations* New York: Academic Press.
- Andreev, E.M. and Kingkade, W.W., 2015. Average age at death in infancy and infant mortality level: Reconsidering the Coale-Demeny formulas at current levels of low mortality. *Demographic Research*, 33, pp.363-390.
- Human Mortality Database Methods Protocol.
- Tools for Demographic Estimation.

See Also

- `ex_to_lifetab_brass()` Calculate life table from minimal inputs
- `q0_to_m0()` Convert between infant mortality measures
- `tfr()` Calculate total fertility rate

Examples

```
library(dplyr)

## life table for females based on 'level 1'
## mortality rates "West" model life table
west_lifetab |>
```

```
    filter(sex == "Female",
           level == 1) |>
    lifetab(mx = mx)

## change method for infant and children from
## default ("constant") to "CD"
west_lifetab |>
    filter(sex == "Female",
           level == 1) |>
    lifetab(mx = mx,
            sex = sex,
            infant = "CD",
            child = "CD")

## calculate life expectancies
## for all levels, using the 'by'
## argument to distinguish levels
west_lifetab |>
    lifeexp(mx = mx,
            sex = sex,
            infant = "CD",
            child = "CD",
            by = level)

## obtain the same result using
## 'group_by'
west_lifetab |>
  group_by(level) |>
  lifeexp(mx = mx,
          sex = sex,
          infant = "CD",
          child = "CD")

## calculations based on 'qx'
west_lifetab |>
  lifeexp(qx = qx,
          sex = sex,
          by = level)

## life expectancy at age 60
west_lifetab |>
  filter(level == 10) |>
  lifeexp(mx = mx,
          at = 60,
          sex = sex)

## life expectancy at ages 0 and 60
west_lifetab |>
  filter(level == 10) |>
  lifeexp(mx = mx,
          at = c(0, 60),
          sex = sex)
```

`logit` *Logit and Inverse-Logit Functions*

Description

Transform values to and from the logit scale. `logit()` calculates

Usage

```
logit(p)

invlogit(x)
```

Arguments

`p`	Values in the interval `[0, 1]`. Can be an atomic vector, a matrix, or an rvec.
`x`	Values in the interval `(-Inf, Inf)`. Can be an atomic vector, a matrix, or an rvec.

Details

$$x = \log \left(\frac{p}{1-p} \right)$$

and `invlogit()` calculates

$$p = \frac{e^x}{1 + e^x}$$

To avoid overflow, `invlogit()` uses $p = \frac{1}{1+e^{-x}}$ internally for x where $x > 0$.

In some of the demographic literature, the logit function is defined as

$$x = \frac{1}{2} \log \left(\frac{p}{1-p} \right).$$

`logit()` and `invlogit()` follow the conventions in statistics and machine learning, and omit the $\frac{1}{2}$.

Value

- A vector of doubles, if `p` or `x` is a vector.
- A matrix of doubles, if `p` or `x` is a matrix.
- An object of class `rvec_dbl`, if `p` or `x` is an rvec.

Examples

```
p <- c(0.5, 1, 0.2)
logit(p)
invlogit(logit(p))
```

---

matrix_to_list_of_cols

Turn a Matrix Into a List of Columns or Rows

Description

Given a matrix, create a list, each element of which contains a column or row from the matrix.

Usage

```
matrix_to_list_of_cols(m)

matrix_to_list_of_rows(m)
```

Arguments

m	A matrix

Details

`matrix_to_list_of_cols()` and ‘matrix_to_list_of_rows() are internal functions, for use by developers, and would not normally be called directly by end users.

Value

- `matrix_to_list_of_cols()` A list of vectors, each of which is a column from `x`.
- `matrix_to_list_of_rows()`, A list of vectors, each of which is a row from `x`.

Examples

```
m <- matrix(1:12, nrow = 3)
matrix_to_list_of_cols(m)
matrix_to_list_of_rows(m)
```

nzmort *Mortality Data for New Zealand*

Description

Counts of deaths and population, by age, sex, and calendar year, plus mortality rates, for New Zealand, 2021-2022.

Usage

```
nzmort
```

Format

A data frame with 84 rows and the following variables:

- `year`: Calendar year.
- `gender`: `"Female"`, and `"Male"`.
- `age`: Age, in life table age groups, with an open age group of 95+.
- `deaths`: Counts of deaths, randomly rounded to base 3.
- `popn`: Estimates of average annual population.
- `mx`: Mortality rates (deaths / popn).

Source

Modified from data in tables "Deaths by age and sex (Annual-Dec)" and "Estimated Resident Population by Age and Sex (1991+) (Annual-Dec)" from Stats NZ online database *Infoshare*, downloaded on 24 September 2023.

nzmort_rvec *Mortality Data and Probabilistic Rates for New Zealand*

Description

A modified version of `link{nzmort}` where `mx` columns is an rvec, rather than an ordinary R vector. The rvec holds the random draws from the posterior distribution obtained from by a Bayesian statistical model.

Usage

```
nzmort_rvec
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 84 rows and 4 columns.

q0_to_m0 — *Convert q0 to m0*

Description

Convert the probability of dying during infancy (q0) to the mortality rate for infancy (m0).

Usage

```
q0_to_m0(
  q0,
  sex = NULL,
  a0 = NULL,
  infant = c("constant", "linear", "CD", "AK")
)
```

Arguments

`q0`	Probability of dying in first year of life. A numeric vector or an rvec.
`sex`	Biological sex. A vector the same length as `q0`, with labels that can be interpreted by `reformat_sex()`. Needed only when `infant` is `"CD"` or `"AK"`.
`a0`	Average age at death for infants who die. Optional. See help for `lifetab()`.
`infant`	Calculation method. See help for `lifetab()`. Default is `"constant"`.

Value

A numeric vector or rvec.

Warning

The term "infant mortality rate" is ambiguous. Demographers sometimes use it to refer to m0 (which is an actual rate) and sometimes use it to refer to q0 (which is a probability.)

See Also

- `lifetab()` Calculate a full life table.

Examples

```
library(dplyr, warn.conflicts = FALSE)
west_lifetab |>
 filter(age == 0, level <= 5) |>
 select(level, sex, age, mx, qx) |>
 mutate(m0 = q0_to_m0(q0 = qx, sex = sex, infant = "CD"))
```

reformat_age *Reformat Age Group Labels*

Description

Convert age group labels to one of three formats:

- Single-year age groups, eg "0", "1", ..., "99", "100+".
- Life table age groups, eg "0", "1-4", "5-9", ..., "95-99", "100+"'.
- Five-year age groups, eg "0-4", "5-9", ..., "95-99", "100+".

By default `reformat_age()` returns a factor that includes all intermediate age groups. See below for examples.

Usage

```
reformat_age(x, factor = TRUE)
```

Arguments

`x`	A vector.
`factor`	Whether the return value should be a factor.

Details

`reformat_age()` applies the following algorithm:

1. Tidy and translate text, eg convert "20 to 24 years" to "20-24", convert "infant" to "0", or convert "100 or more" to "100+".
2. Check whether the resulting labels could have been produced by `age_labels()`. If not, throw an error.
3. If `factor` is `TRUE` (the default), then return a factor. The levels of this factor include all intermediate age groups. Otherwise return a character vector.

When `x` consists entirely of numbers, `reformat_age()` also checks for two special cases:

- If every element of `x` is a multiple of 5, and if `max(x) >= 50`, then `x` is assumed to describe 5-year age groups
- If every element of `x` is 0, 1, or a multiple of 5, with `max(x) >= 50`, then `x` is assumed to describe life table age groups.

Value

If `factor` is `TRUE`, then `reformat_age()` returns a factor; otherwise it returns a character vector.

See Also

`age_labels()`, `reformat_sex()`

Examples

```
reformat_age(c("80 to 84", "90 or more", "85 to 89"))

## factor contains intermediate level missing from 'x'
reformat_age(c("80 to 84", "90 or more"))

## non-factor
reformat_age(c("80 to 84", "90 or more"),
          factor = FALSE)

## single
reformat_age(c("80", "90plus"))

## life table
reformat_age(c("0",
            "30-34",
            "10--14",
            "1-4 years"))
```

---

reformat_sex — *Reformat a Binary Sex Variable*

Description

Reformat a binary sex variable so that it consists entirely of values "Female", "Male", and possibly NA and any values included in except.

Usage

```
reformat_sex(x, except = NULL, factor = TRUE)
```

Arguments

x	A vector.
except	Values to exclude when reformatting.
factor	Whether the return value should be a factor.

Details

When parsing labels, reformat_sex() ignores case: "FEMALE" and "fEmAlE" are equivalent.

White space is removed from the beginning and end of labels.

reformat_sex() does not try to interpreting numeric codes (eg 1, 2).

Value

If factor is TRUE, then reformat_age() returns a factor; otherwise it returns a character vector.

See Also

age_labels(), reformat_age()

Examples

```
reformat_sex(c("F", "female", NA, "MALES"))

## values supplied for 'except'
reformat_sex(c("Fem", "Other", "Male", "M"),
             except = c("Other", "Diverse"))

## return an ordinary character vector
reformat_sex(c("F", "female", NA, "MALES"),
             factor = FALSE)
```

---

rr3 — *Randomly Round A Vector of Integers to Base 3*

Description

Apply the 'Random Round to Base 3' (RR3) algorithm to a vector of integers (or doubles where `round(x) == x`.)

Usage

```
rr3(x)
```

Arguments

x	A vector of integers (in the sense that `round(x) == x`.) Can be an rvec.

Details

The RR3 algorithm is used by statistical agencies to confidentialize data. Under the RR3 algorithm, an integer n is randomly rounded as follows:

- If n is divisible by 3, leave it unchanged
- If dividing n by 3 leaves a remainder of 1, then round down (subtract 1) with probability 2/3, and round up (add 2) with probability 1/3.
- If dividing n by 3 leaves a remainder of 1, then round down (subtract 2) with probability 1/3, and round up (add 1) with probability 2/3.

RR3 has some nice properties:

- The randomly-rounded version of n has expected value n.
- If n non-negative, then the randomly rounded version of n is non-negative.
- If n is non-positive, then the randomly rounded version of n is non-positive.

Value

A randomly-rounded version of x.

Examples

```
x <- c(1, 5, 2, 0, -1, 3, NA)
rr3(x)
```

---

set_age_open *Specify Open Age Group*

Description

Set the lower limit of the open age group. Given a vector of age group labels, recode all age groups with a lower limit greater than or equal to <lower> to <lower>+.

Usage

```
set_age_open(x, lower)
```

Arguments

x	A vector of age labels.
lower	An integer. The lower limit for the open age group.

Details

set_age_open() requires that x and the return value have a a five-year, single-year, or life table format, as described in age_labels().

Value

A modified version of x.

See Also

- set_age_open() uses age_lower() to identify lower limits
- age_labels() for creating age labels from scratch

Examples

```
x <- c("100+", "80-84", "95-99", "20-24")
set_age_open(x, 90)
set_age_open(x, 25)
```

tfr — *Calculate Total Fertility Rates*

Description

Calculate the total fertility rate (TFR) from age-specific fertility rates.

Usage

```
tfr(
  data,
  asfr = NULL,
  age = age,
  sex = NULL,
  by = NULL,
  denominator = 1,
  suffix = NULL
)
```

Arguments

data	Data frame with age-specific fertility rates and age
asfr	Age-specific fertility rates. Possibly an rvec.
age	<tidyselect> Age group labels. The labels must be interpretable by functions such as reformat_age() and age_group_type(). The age groups must not have gaps, and the highest age group must be "closed" (ie have an upper limit.)
sex	<tidyselect> Sex/gender of the child (not the parent).
by	<tidyselect> Separate total fertility rates are calculated for each combination the by variables. If data is a grouped data frame, then the grouping variables take precedence over by.
denominator	The denominator used to calculate asfr. Default is 1.
suffix	Optional suffix added to "tfr" column in result.

Details

The total fertility rate is a summary measures for current fertility levels that removes the effect of age structure. Is obtained by summing up age-specific fertility rates, multiplying each rate by the width of the corresponding age group. For instance, the rate for age group "15-19" is multiplied by 5, and the rate for age group "15" is multiplied by 1.

The total fertility rate can be interpreted as the number of average children that a person would have, under prevailing fertility rates, if the person survived to the maximum age of reproduction. The hypothetical person is normally a woman, since age-specific fertility rates normally use person-years lived by women as the denominator. But it can apply to men, if the age-specific fertility rates are "paternity rates", ie rates that use person-years lived by men as the denominator.

Value

A tibble.

Sex-specific fertility rates

Age-specific fertility rates do not normally specify the sex of the children who are born. In cases where they do, however, rates have to be summed across sexes to give the total fertility rates. If `tfr()` is supplied with a `sex` argument, it assumes that `sex` applies to the births, and sums over the sexes.

Denominator

Published tables of age-specific fertility rates often express the rates as births per 1000 person-years lived, rather than per person-year lived. (Sometimes this is expressed as "births per 1000 women".) In these cases

Using rvecs to represent uncertainty

An rvec is a 'random vector', holding multiple draws from a distribution. Using an rvec for the `asfr` argument to `tfr()` is a way of representing uncertainty. This uncertainty is propagated through to the TFR, which will also be rvecs.

See Also

- `lifeexp()` Calculate life expectancy from age-specific mortality rates.

Examples

```
iran_fertility |>
  tfr(asfr = rate,
      by = c(area, time),
      denominator = 1000)
```

`to_matrix` *Build a Matrix from Measure and ID Variables*

Description

Build a matrix where the elements are values of a measure variable, and the rows and columns are formed by observed combinations of ID variables. The ID variables picked out by `rows` and `cols` must uniquely identify cells. `to_matrix()`, unlike `stats::xtabs()`, does not sum across multiple combinations of ID variables.

Usage

```
to_matrix(x, rows, cols, measure)
```

Arguments

`x`	A data frame.
`rows`	The ID variable(s) used to distinguish rows in the matrix.
`cols`	The ID variable(s) used to distinguish columns in the matrix.
`measure`	The measure variable, eg rates or counts.

Value

A matrix

Examples

```
x <- expand.grid(age = c(0, 1, 2),
                 sex = c("F", "M"),
                 region = c("A", "B"),
                 year = 2000:2001)
x$count <- 1:24

to_matrix(x,
          rows = c(age, sex),
          cols = c(region, year),
          measure = count)

to_matrix(x,
          rows = c(age, sex, region),
          cols = year,
          measure = count)

## cells not uniquely identified
try(
to_matrix(x,
          rows = age,
          cols = sex,
          measure = count)
)
```

`trim_01` *Trim Values So They Are Between 0 and 1*

Description

Trim a vector so that all values are greater than 0 and less than 1.

Usage

```
trim_01(x)
```

Arguments

`x` A numeric vector. Can be an rvec.

Details

If

- `min` is lowest element of `x` that is higher than 0, and
- `max` is the highest element of `x` that is lower than 1, then `trim_01()`
- shifts all elements of `x` that are lower than `min` upwards, so that they equal `min`, and
- shifts all elements of `x` that are higher than `max` downwards, so that they equal `max`.

Value

A trimmed version of `x`

See Also

- `logit()`, `invlogit()` Logit transformation

Examples

```
x <- c(1, 0.98, -0.001, 0.5, 0.01)
trim_01(x)
```

`west_lifetab` *Coale-Demeny West Model Life Tables*

Description

Life table quantities from the "West" family of Coale-Demeny model life tables.

Usage

```
west_lifetab
```

Format

A data frame with 1,050 rows and the following variables:

- `level`: Index for life table. Lower level implies lower life expectancy.
- `sex`: `"Female"`, and `"Male"`.
- `age`: Age, in life table age groups, with an open age group of 95+.
- `mx`: Mortality rate.
- `ax`: Average years lived in age interval by people who die in that interval.
- `qx`: Probability some alive at start of age interval dies during interval.

- `lx`: Number of people still alive at start of age interval.
- `dx`: Number of people dying during age interval.
- `Lx`: Number of person-years lived during age interval.
- `ex`: Expectation of life at start of age interval.

Source

Coale A, Demeny P, and Vaughn B. 1983. Regional model life tables and stable populations. 2nd ed. New York: Academic Press, accessed via `demogR::cdmltw()`.

Index