Package 'canregtools'

November 26, 2025

Title Analyzing Population-Based Cancer Registry Data

Version 0.2.10

Description Tools for cleaning, analyzing, visualizing, and reporting data from Population-Based Cancer Registries (PBCRs), with standardized workflows for filtering, selecting, and restructuring data, designed for routine registry operations.

License MIT + file LICENSE

URL https://github.com/gigu003/canregtools, https://gigu003.github.io/canregtools/, https://canregtools.chenq.site

BugReports https://github.com/gigu003/canregtools/issues

Depends R (>= 4.1.0)

Imports dplyr, graphics, grDevices, readxl, rlang, tidyr, purrr, stats

Suggests covr, knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

VignetteBuilder knitr

NeedsCompilation no

Author Qiong Chen [aut, cre, cph] (ORCID: <https://orcid.org/0000-0003-2401-0046>, affiliation: Department of Cancer Epidemiology, The Affiliated Cancer Hospital of Zhengzhou University & Henan Cancer Hospital, Henan Cancer Registry in China)

Maintainer Qiong Chen <chenq08@126.com>

Contents

`canregtools-package` *canregtools: Analyzing Population-Based Cancer Registry Data*

Description

Tools for cleaning, analyzing, visualizing, and reporting data from Population-Based Cancer Registries (PBCRs), with standardized workflows for filtering, selecting, and restructuring data, designed for routine registry operations.

Author(s)

Maintainer: Qiong Chen <chenq08@126.com> (ORCID) (Department of Cancer Epidemiology, The Affiliated Cancer Hospital of Zhengzhou University & Henan Cancer Hospital, Henan Cancer Registry in China) [copyright holder]

See Also

Useful links:

- https://github.com/gigu003/canregtools
- https://gigu003.github.io/canregtools/
- https://canregtools.chenq.site
- Report bugs at https://github.com/gigu003/canregtools/issues

---

`add_labels` *Add variable labels for data set.*

Description

`add_labels()` adds labels for selected variables such as `sex`, `cancer`, and `areacode` in the dataset based on the specified language and label type.

Usage

```
add_labels(
  x,
  vars = c("sex", "cancer", "areacode"),
  names = NULL,
  label_type = "full",
  lang = "zh",
  sep = " ",
  as_factor = TRUE
)
```

Arguments

x	A data frame or tibble to be labeled.
vars	A character vector of variable names to label. Default includes "sex", "cancer", and "areacode".
names	A character vector contains the labels added.
label_type	Type of the label used ("full" or "abbr").
lang	Character, specify the output language, options are 'cn', or 'en', default is 'cn'.
sep	String used to separate labels when multiple languages are specified.
as_factor	Logical, indicate whether output value as factor.

Value

A data frame or tibble with labeled variables and reordered columns.

Examples

```
data("canregs")
asr <- create_asr(canregs[[1]], year, sex, cancer)
asr <- add_labels(asr, label_type = "full", lang = "zh")
asr <- add_labels(asr, label_type = "full", lang = "en")
```

---

add_var_labels *Add Variable Labels with Units*

Description

Returns formatted labels (with optional units) for commonly used statistical variables in cancer registry reporting, such as incidence, mortality, crude rate, age-standardized rate, and proportion.

Usage

```
add_var_labels(x, label_type = "abbr", lang = "cn", break_line = TRUE)
```

Arguments

x	A character vector of variable codes to be labeled (e.g., "cr", "asr", "prop").
label_type	Label style: "abbr" (default) for abbreviation, "full" for full name.
lang	Language for the labels. One of "cn" (default) for Chinese, "en" for English. Special options "code" or "icd10" return the variable code or ICD-10 code instead.
break_line	Logical. If TRUE (default), adds a line break (\\n) before the unit. If FALSE, appends the unit directly after the label, suitable for HTML rendering (e.g., in flextable).

Details

This function looks up the corresponding label for each input variable code in the internal dictionary tidy_var_maps[["stats"]], and optionally adds a unit suffix (e.g., (1/10⁵) or (%)) depending on the variable type. It supports Chinese and English, full names or abbreviations, and can format the unit with or without a line break.

Value

A character vector of the same length as x, where each element is a formatted label.

Examples

```
add_var_labels(c("cr", "asr"))
add_var_labels(c("cr", "prop"), label_type = "full", lang = "en")
add_var_labels("asr", break_line = FALSE)
```

---

ageadjust *Calculate the age standardized rate using the direct method*

Description

This function calculates the **age-standardized rate (ASR)** using the **direct method** of standardization. It allows for adjustment of crude disease or mortality rates to a standard population structure, enabling valid comparisons across populations with different age distributions. The function also supports computation of confidence intervals using gamma, normal, or log-normal methods, and returns both crude rate (CR) and ASR with associated variances and interval estimates.

Usage

```
ageadjust(
  count,
  pop,
  rate = NULL,
  stdpop = NULL,
  method = "gamma",
  conf_level = 0.95,
  mp = 1e+05
)
```

Arguments

count	The number of cases of a specific disease or condition.
pop	The total population of the same group or region where the disease cases (count) were observed.
rate	Disease rate, which is the number of cases (count) per unit of population (pop).
stdpop	Standardized population for age standardization.
method	Method used for calculating the age-standardized rate, options are 'gamma', 'normal', or 'lognormal', default is 'gamma'.
conf_level	Confidence level for calculating confidence intervals, value between 0 and 1, default is 0.95.
mp	A multiplier used to scale the calculated rates. Default is 100000.

Value

Age standardized rate and its confidence interval.

Examples

```
cases <- c(50, 60, 45, 70)
pop <- c(1000, 1200, 1100, 900)
spop <- c(800, 1000, 1100, 900)
ageadjust(cases, pop, stdpop = spop, mp = 100000)
```

`calc_age` *Calculate the actual age in completed years between two dates*

Description

Computes the exact age in full years at the time of an event (e.g., diagnosis, death, or survey) by comparing a person's birth date to the date of the event.

Usage

```
calc_age(birth_date, onset_date)
```

Arguments

`birth_date`	A vector of birth dates in `Date` format.
`onset_date`	A vector of corresponding event dates in `Date` format (e.g., date of diagnosis).

Details

This function calculates age in *completed years*, taking into account whether the birthday has occurred before the event date in the given year. If `birth_date` or `onset_date` contains `NA`, the corresponding result will be `NA`.

Value

A numeric vector of ages in years, with the same length as `birth_date` and `onset_date`.

Examples

```
# Generate random birth dates
set.seed(123)
sdate <- as.Date("1960-01-01")
edate <- as.Date("1980-12-31")
bdate <- sample(seq(sdate, edate, by = "1 day"), 100, replace = TRUE)

# Generate random event dates
sdate <- as.Date("2020-01-01")
edate <- as.Date("2023-07-08")
event <- sample(seq(sdate, edate, by = "1 day"), 100, replace = TRUE)

# Calculate ages
ages <- calc_age(bdate, event)
head(ages)
# Handle missing values
bdate[1] <- NA
event[2] <- NA
calc_age(bdate, event)[1:5]
```

canregs *Example Population-Based Cancer Registry Data (canregs)*

Description

This dataset provides example data from population-based cancer registries (PBCRs), structured using the canregs class. It can be used for demonstration, testing, or development purposes within the canregtools package.

Usage

```
canregs
```

Format

A list of PBCR datasets with class canregs.

Source

Henan Province Cancer Registry, China.

Examples

```
data("canregs")
summary(canregs[[1]])
```

classify_areacode *Classify codes for the administrative divisions of China*

Description

Categorizes six-digit administrative division codes of the People's Republic of China (as per GB/T 2260-2007) into several structured components, including province, city, area type (urban/rural), registry code, and regional classification.

Usage

```
classify_areacode(x)
```

Arguments

x	A vector of six-digit Chinese administrative area codes, either as numeric or character strings.

Details

This function standardizes and validates area codes, identifies their administrative levels, and attaches metadata used in cancer registration systems. It also supports external dictionaries (from the canregtools configuration folder) to provide more accurate classification of area types and registry mapping. Classify Codes for the administrative divisions of the People's Republic of China(GB/T 2260-2007) into different categories, including 'province', city', 'area_type', and 'registry' attributes.

Value

A list with the following named elements:

- `areacode`: Validated area codes. Invalid entries are replaced with `NA`.
- `registry`: Registry codes corresponding to each area, using a built-in or cached dictionary.
- `province`: Province-level codes formed by taking the first two digits and appending `"0000"`.
- `city`: City-level codes formed by taking the first four digits and appending `"00"`.
- `area_type`: Urban-rural classification codes: `"910000"` for urban, `"920000"` for rural. This can be updated using `write_registry()` function which stored the dictionary in (`area_type_dict.rds`)
- `region`: Region classification codes derived from province codes, ending in `"0000"`.

See Also

`write_registry()`

Examples

```
classify_areacode(c("110000", "320500", "440300"))
```

---

classify_areacode2 *Classify codes for the administrative divisions of China*

Description

Categorizes six-digit administrative division codes of the People's Republic of China (as per GB/T 2260-2007) into several structured components, including province, city, area type (urban/rural), registry code, and regional classification.

Usage

```
classify_areacode2(x, attr = "registry")
```

Arguments

`x`	A vector of six-digit Chinese administrative area codes, either as numeric or character strings.
`attr`	A character vector of attributes to return. Options include: • `"registry"`: Cancer registry codes. • `"province"`: Province-level codes. • `"city"`: City-level codes. • `"area_type"`: Urban/rural classification. • `"region"`: Regional classification based on province. • Any other string will be treated as a custom dictionary name written using `write_registry()`.

Details

This function standardizes and validates area codes, identifies their administrative levels, and attaches metadata used in cancer registration systems. It also supports external dictionaries (from the canregtools configuration folder) to provide more accurate classification of area types and registry mapping. Classify Codes for the administrative divisions of the People's Republic of China(GB/T 2260-2007) into different categories, including 'province', city', 'area_type', and 'registry' attributes.

Value

A list with the following named elements:

- areacode: Validated area codes. Invalid entries are replaced with NA.
- registry: Registry codes corresponding to each area, using a built-in or cached dictionary.
- province: Province-level codes formed by taking the first two digits and appending "0000".
- city: City-level codes formed by taking the first four digits and appending "00".
- area_type: Urban-rural classification codes: "910000" for urban, "920000" for rural. This can be updated using write_registry() function which stored the dictionary in (area_type_dict.rds)
- region: Region classification codes derived from province codes, ending in "0000".

See Also

write_registry()

Examples

```
classify_areacode(c("110000", "320500", "440300"))
```

---

classify_childhood *Classify childhood cancer according to the ICCC3 standards*

Description

classify_childhood()classifies childhood cancer based on ICD-O-3 codes, which include topography, morphology, and behavior codes, using the International Classification of Childhood Cancer, Third Edition (ICCC3).

Usage

```
classify_childhood(topo, morp, beha, type = "sub", version = "v2005")
```

Arguments

topo	A character vector of ICD-O-3 topography codes (e.g., "C15.6" or "C156").
morp	A character vector of ICD-O-3 morphology codes (e.g., "8000" or "M8140").
beha	A numeric or character vector representing ICD-O-3 behavior codes.
type	A string specifying the type of classification to return: "main" for main groups or "sub" for subgroups. Defaults to "main".
version	A string specifying the version of the ICCC-3 rules to use: either "v2005" or "v2017". Defaults to "v2005".

Value

A numeric vector of ICCC-3 classification codes. If `type = "sub"`, returns subgroup codes; if `type = "main"`, returns main group codes.

References

Steliarova-Foucher, E., Stiller, C., Lacour, B. and Kaatsch, P. (2005), International Classification of Childhood Cancer, third edition†‡. Cancer, 103: 1457-1467. doi:10.1002/cncr.20910

Examples

```
topo <- c("C15.2", "C16.2", "C34.2")
morp <- c("8000", "8040", "8170")
beha <- c("3", "3", "3")
child_code <- classify_childhood(topo, morp, beha, type = "main")
```

---

classify_icd10 *Classify ICD10 codes to cancer categories*

Description

Classify ICD10 codes into Cancer Categories according to the specified category type and language.

Usage

```
classify_icd10(
  x,
  cancer_type = "big",
  lang = "code",
  label_type = "abbr",
  as_factor = FALSE
)
```

Arguments

x	The ICD10 codes of cancer part ('C00-C98 and D00-D48') which used in by the Population-Based Cancer Registration ('PBCR').
cancer_type	A character string specifying the classification method used to categorize ICD-10 codes. This determines how ICD-10 codes are classified. Options include `"big"` (classify ICD-10 codes into 26 cancer categories), `"small"` (classify ICD-10 codes into 59 cancer categories, more specific categories), `"system"` (classify ICD-10 codes into organ system), and `"gco"` (classify ICD-10 code into cancer categories same as classification published by the Global Cancer Observatory). This parameter is only available when the input data is a vector of ICD-10 codes, or object with class of `'canreg'` or `'canregs'`.
lang	Character, specify the output language, options are 'cn', or 'en', default is 'cn'.
label_type	Type of the label used ("full" or "abbr").
as_factor	Logical, indicate whether output value as factor.

Value

Cancer code.

Examples

```
icd10 <- c("C15.2", "C33.4", "C80.9", "C26.2", "C16.3")
classify_icd10(icd10, cancer_type = "big")
classify_icd10(icd10, cancer_type = "small")
classify_icd10(icd10, cancer_type = "system")
classify_icd10(icd10, cancer_type = "gco")
```

---

classify_morp *Classify ICD-O-3 morphology codes into categories*

Description

Classify ICD-O-3 morphology codes into categories

Usage

```
classify_morp(x)
```

Arguments

x	Character vector of ICD-O-3 morphology codes. Values not present in the internal dictionary will be returned as NA.

Value

Character vector of categories corresponding to x.

Examples

```
morps <- c("8140","8070","8050","8051","9900","9800","9993")
classify_morp(morps)
```

---

classify_topo *Classify ICD-O-3 topography codes into categories*

Description

Classify ICD-O-3 topography codes into categories

Usage

```
classify_topo(x, cancer_type = "big")
```

Arguments

x	Character vector of ICD-O-3 topography codes. Values not present in the internal dictionary will be returned as NA.
cancer_type	Cancer type.

Value

Character vector of categories corresponding to x.

---

combine_tp *Sum named numeric vectors*

Description

Take a list of named numeric vectors and sum values with the same names

Usage

```
combine_tp(object)
```

Arguments

object	A list where each element is a named numeric vector

Value

A named integer vector with names sorted alphabetically and values summed across vectors.

---

count_canreg *Count and classify* canreg *data*

Description

The count_canreg() function is a generic method used to summarize population-based cancer registry data. It supports both single (canreg) and multiple (canregs) registry objects. The function aggregates cancer cases by age group and classifies cancer types using standardized coding systems.

Usage

```
count_canreg(
  x,
  age_breaks = c(0, 1, seq(5, 85, 5)),
  label_tail = NULL,
  cancer_type = "big"
)

## S3 method for class 'canregs'
count_canreg(x, ...)
```

```
## S3 method for class 'canreg'
count_canreg(
  x,
  age_breaks = c(0, 1, seq(5, 85, 5)),
  label_tail = NULL,
  cancer_type = "big"
)
```

Arguments

x	Object with class of 'canreg' or 'canregs'.
age_breaks	A numeric vector specifying the breakpoints for age grouping. Defaults to `c(0, 1, seq(5, 85, 5))`.
label_tail	Optional. A string to append to age group labels (e.g., `"+"` for open-ended intervals). Defaults to `NULL`.
cancer_type	A character string specifying the classification method used to categorize ICD-10 codes. This determines how ICD-10 codes are classified. Options include `"big"` (classify ICD-10 codes into 26 cancer categories), `"small"` (classify ICD-10 codes into 59 cancer categories, more specific categories), `"system"` (classify ICD-10 codes into organ system), and `"gco"` (classify ICD-10 code into cancer categories same as classification published by the Global Cancer Observatory). This parameter is only available when the input data is a vector of ICD-10 codes, or object with class of `'canreg'` or `'canregs'`.
...	Additional arguments passed to the method for individual `canreg` objects.

Value

Object with class of 'fbswicd' or 'fbswicds'.

See Also

`cr_clean()`, `classify_icd10()`, `create_asr()`

Examples

```
data("canregs")
fbsw <- count_canreg(canregs, age_breaks = c(0, 15, 65), cancer_type = "big")
fbsw <- count_canreg(canregs, cancer_type = "gco")

# Count object with class of `canregs`
fbsw <- count_canreg(canregs, cancer_type = "small")

# Count object with class of `canreg`
fbsw <- count_canreg(canregs[[1]], cancer_type = "big")
```

create_age_rate *Calculate age specific rate*

Description

create_age_rate() computes age-specific rates from object with class of canreg,fbswicd, or canregs, fbswicds. It calculates the rates for specified events (e.g., fbs) across age groups, stratified by variables such as year, sex, or cancer type.

Usage

```
create_age_rate(
  x,
  ...,
  event = "fbs",
  cancer_type = "big",
  format = "long",
  mp = 1e+05,
  decimal = 6,
  show_pop = FALSE,
  collapse = TRUE
)

## S3 method for class 'canreg'
create_age_rate(x, ..., cancer_type = "big")

## S3 method for class 'canregs'
create_age_rate(x, ..., cancer_type = "big", collapse = TRUE)

## S3 method for class 'fbswicds'
create_age_rate(
  x,
  ...,
  event = "fbs",
  format = "long",
  mp = 1e+05,
  decimal = 6,
  show_pop = FALSE,
  collapse = TRUE
)

## S3 method for class 'fbswicd'
create_age_rate(
  x,
  ...,
  event = "fbs",
  format = "long",
  mp = 1e+05,
  decimal = 6,
  show_pop = FALSE
)
```

Arguments

x	The input data, object with class of 'fbswicd', 'fbswicds', 'canreg', or 'canregs'.
...	One or more variables used for stratification. For example, you can stratify by sex, year, cancer, or just by year. If sex is not passed as a parameter, the output will be the result for the combined gender.
event	A variable used to specify the type of calculation, options are "fbs" or "sws", "fbs" for cancer incidence, and "sws" for cancer mortality.
cancer_type	A character string specifying the classification method used to categorize ICD-10 codes. This determines how ICD-10 codes are classified. Options include "big" (classify ICD-10 codes into 26 cancer categories), "small" (classify ICD-10 codes into 59 cancer categories, more specific categories), "system" (classify ICD-10 codes into organ system), and "gco" (classify ICD-10 code into cancer categories same as classification published by the Global Cancer Observatory). This parameter is only available when the input data is a vector of ICD-10 codes, or object with class of 'canreg' or 'canregs'.
format	Format of the output data frame, either "long" or "wide".
mp	A constant to multiply rates by (e.g. mp=1000 for rates per 1000).
decimal	This parameter specifies the number of decimal places to round the results. The default is 2, which means rates will be rounded to two decimal places.
show_pop	Logical value whether output population or not.
collapse	Logical value whether output result as age_rate or age_rates.

Value

A data frame of age-specific rates.

Examples

```
data("canregs")
agerate <- create_age_rate(canregs, year, sex, cancer)

data <- canregs[[1]]
agerate <- create_age_rate(data, year, sex)

agerate <- create_age_rate(canregs, year, cancer_type = "system")

fbsws <- count_canreg(canregs)
agerate <- create_age_rate(fbsws, year, sex)

data <- canregs[[2]]
fbsw <- count_canreg(data, cancer_type = "small")
agerate <- create_age_rate(fbsw, year, sex, cancer)
```

create_asr *Calculate age-standardized rate (ASR)*

Description

`create_asr()` calculate age-standardized rates (ASRs) from object with class of `canreg`, `canregs`, `fbswicd`, or `fbswicds`. It supports stratification by multiple variables, allows the specification of different standard population structures, and provides flexibility in the inclusion of variance, confidence intervals, and population data.

Usage

```
create_asr(
  x,
  ...,
  event = "fbs",
  std = c("cn2000", "wld85"),
  cancer_type = "big",
  mp = 1e+05,
  decimal = 2,
  show_var = FALSE,
  show_ci = FALSE,
  collapse = TRUE
)

## S3 method for class 'canregs'
create_asr(x, ..., cancer_type = "big", collapse = TRUE)

## S3 method for class 'canreg'
create_asr(x, ..., cancer_type = "big")

## S3 method for class 'fbswicds'
create_asr(
  x,
  ...,
  event = "fbs",
  std = c("cn2000", "wld85"),
  mp = 1e+05,
  decimal = 2,
  show_pop = FALSE,
  show_var = FALSE,
  show_ci = FALSE,
  collapse = TRUE
)

## S3 method for class 'fbswicd'
create_asr(
  x,
  ...,
  event = "fbs",
  std = c("cn2000", "wld85"),
```

```
  mp = 1e+05,
  decimal = 2,
  show_pop = FALSE,
  show_var = FALSE,
  show_ci = FALSE
)
```

Arguments

x	The input data, object with class of 'fbswicd', 'fbswicds', 'canreg', or 'canregs'.
...	One or more variables used for stratification. For example, you can stratify by sex, year, cancer, or just by year. If sex is not passed as a parameter, the output will be the result for the combined gender.
event	A variable used to specify the type of calculation, options are "fbs" or "sws", "fbs" for cancer incidence, and "sws" for cancer mortality.
std	Specify the standard population structure in the 'std_pop' data frame used for calculating standardized rates. When calculating standardized rates for multiple standard populations, specify std = c(segi, china).
cancer_type	A character string specifying the classification method used to categorize ICD-10 codes. This determines how ICD-10 codes are classified. Options include "big" (classify ICD-10 codes into 26 cancer categories), "small" (classify ICD-10 codes into 59 cancer categories, more specific categories), "system" (classify ICD-10 codes into organ system), and "gco" (classify ICD-10 code into cancer categories same as classification published by the Global Cancer Observatory). This parameter is only available when the input data is a vector of ICD-10 codes, or object with class of 'canreg' or 'canregs'.
mp	A constant to multiply rates by (e.g. mp=1000 for rates per 1000).
decimal	This parameter specifies the number of decimal places to round the results. The default is 2, which means rates will be rounded to two decimal places.
show_var	Logical value whether output variance or not.
show_ci	Logical value whether output confidence(lower or upper bound) or not.
collapse	Logical value whether output result as asr or asrs.
show_pop	Logical value whether output population or not.

Value

A data frame or tibble contains the age standard rates and CIs.

See Also

ageadjust for age-adjusted rate calculations. truncrate for truncated rate calculations.

Examples

```
data("canregs")
asr_inci <- create_asr(canregs, event = "fbs", year, sex, cancer)
asr_mort <- create_asr(canregs, event = "sws", year, sex, cancer)

# calculate ASR based on object with class of `canregs`
asr <- create_asr(canregs, event = "sws", year, sex, cancer)
```

```
# calculate ASR based on object with class of `canreg`
data <- canregs[[1]]
# calculate ASR using default parameter
asr <- create_asr(data, year, sex, cancer)
head(asr)
# calculate ASR using multiple standard population
asr_multi_std <- create_asr(data, year, sex, cancer,
  std = c("cn82", "cn2000", "wld85")
)
head(asr_multi_std)
# calculate ASR with confidence interval
asr_with_ci <- create_asr(data, year, sex, cancer, show_ci = TRUE)
head(asr_with_ci)
# calculate ASR with population at risk
asr_with_pop <- create_asr(data, year, sex, cancer, show_pop = TRUE)
head((asr_with_pop))
# calculate ASR with variance
asr_with_var <- create_asr(data, year, sex, cancer, show_var = TRUE)
head(asr_with_var)

# calculate ASR based on object with class of `fbswicds`
# convert object with class of `canregs` to object with class of `fbswicds`
fbsws <- count_canreg(canregs)
asrs <- create_asr(fbsws, event = "sws", year, sex, cancer)

# calculate ASR based on object with class of `fbswicd`
# convert object with class of `canreg` to object with class of `fbswicd`
fbsw <- count_canreg(canregs[[1]])
asr <- create_asr(fbsw, event = "sws", year, sex, cancer)
```

`create_quality` *Calculate quality indicators*

Description

`create_quality()`calculate quality indicators from object with class of `canreg`, `canregs`, `fbswicd`, or `fbswicds`. The quality indicators for population-based cancer registries (PBCRs) including:

- `fbs`: Number of incident cases.
- `inci`: Cancer incidence rate.
- `sws`: Number of death cases.
- `mort`: Mortality rate.
- `mv`: Percentage of cases with microscopic verification.
- `mi`: Mortality-to-incidence ratio.
- And other relevant quality metrics for cancer data evaluation.

Usage

```
create_quality(x, ..., decimal = 2, collapse = TRUE)

## S3 method for class 'canreg'
create_quality(x, ..., cancer_type = "big")

## S3 method for class 'canregs'
create_quality(x, ..., cancer_type = "big", collapse = TRUE)

## S3 method for class 'fbswicds'
create_quality(x, ..., decimal = 2, collapse = TRUE)

## S3 method for class 'fbswicd'
create_quality(x, ..., decimal = 2)
```

Arguments

x	The input data, object with class of 'fbswicd', 'fbswicds', 'canreg', or 'canregs'.
...	One or more variables used for stratification. For example, you can stratify by sex, year, cancer, or just by year. If sex is not passed as a parameter, the output will be the result for the combined gender.
decimal	The number of decimal places to include in the resulting quality indicator values. Defaults to 2.
collapse	Logical value whether output result as quality or qualites.
cancer_type	A character string specifying the classification method used to categorize ICD-10 codes. This determines how ICD-10 codes are classified. Options include "big" (classify ICD-10 codes into 26 cancer categories), "small" (classify ICD-10 codes into 59 cancer categories, more specific categories), "system" (classify ICD-10 codes into organ system), and "gco" (classify ICD-10 code into cancer categories same as classification published by the Global Cancer Observatory). This parameter is only available when the input data is a vector of ICD-10 codes, or object with class of 'canreg' or 'canregs'.

Value

A data frame (if applied to a single registry object, 'canreg' or 'fbswicd') or a list of data frames (if applied to a grouped registry object, 'canregs' or 'fbswicds') with a class of either 'quality' or 'qualities'.

Examples

```
data("canregs")
fbsws <- count_canreg(canregs, cancer_type = "system")
qua2 <- create_quality(fbsws, year, sex, cancer)
head(qua2)

# Calculate the quality indicators based on object with class of `canreg`
data <- canregs[[1]]
qua <- create_quality(data, year, sex, cancer, cancer_type = "big")
head(qua)
```

```
# Calculate the quality indicators based on object with class of `canregs`
qua <- create_quality(canregs, year, sex, cancer, cancer_type = "big")
head(qua)

# Calculate the quality indicators based on object with class of `fbswicds`
fbsws <- count_canreg(canregs, cancer_type = "small")
qua <- create_quality(fbsws, year, sex, cancer)
head(qua)

# Calculate the quality indicators based on object with class of `fbswicd`
fbsw <- count_canreg(canregs[[1]], cancer_type = "big")
qua <- create_quality(fbsw, year, sex, cancer)
head(qua)
```

`create_report` *Render standardized cancer registry reports from built-in templates*

Description

The `create_report()` function generates a report from objects with class of `canreg` or `canregs` using pre-defined rmarkdown templates.

Usage

```
create_report(
  data,
  template = "annual",
  title = "Cancer Registry Report",
  output_format = "html_document",
  output_dir = NULL,
  ...
)

## S3 method for class 'canregs'
create_report(
  data,
  template = "annual",
  title = "Cancer Registry Report",
  output_format = "html_document",
  output_dir = NULL,
  ...
)

## S3 method for class 'canreg'
create_report(
  data,
  template = "annual",
  title = "Cancer Registry Report",
  output_format = "html_document",
  output_dir = NULL,
  ...
)
```

Arguments

data	An object of class canreg or canregs.
template	Character string specifying the report template to use. Options include "annual", "quality", or "CI5". Default is "annual".
title	Character. Title of the generated report. Default is "Cancer Registry Report".
output_format	Character. Format of the rendered report. Options are "html_document", "word_document", or "pdf_document". Default is "html_document".
output_dir	Character. Directory where the report will be saved.
...	Additional arguments passed to rmarkdown::render().

Value

No return value; generates a report as a side effect.

Examples

```
## Not run:
data("canregs")
create_report(canregs, template = "quality", title = "QC Report")

## End(Not run)

## Not run:
create_report(canregs, template = "annual", title = "Annual Report")

## End(Not run)

## Not run:
data <- canregs[[1]]
create_report(data, template = "annual", title = "Annual Report")

## End(Not run)
```

---

create_site *Count*

Description

Count

Usage

```
create_site(x, ..., wrap_subsite = FALSE, class_morp = TRUE, drop_nos = TRUE)
```

Arguments

x	Fbswicd
...	Strata
wrap_subsite	Logical
class_morp	Logical
drop_nos	Logical

Value

Data frame

`cr_clean` *Clean canreg data.*

Description

Clean canreg data.

Usage

```
cr_clean(
  x,
  age_breaks = c(0, 1, seq(5, 85, 5)),
  label_tail = NULL,
  cancer_type = "big"
)

## S3 method for class 'canregs'
cr_clean(
  x,
  age_breaks = c(0, 1, seq(5, 85, 5)),
  label_tail = NULL,
  cancer_type = "big"
)

## S3 method for class 'canreg'
cr_clean(
  x,
  age_breaks = c(0, 1, seq(5, 85, 5)),
  label_tail = NULL,
  cancer_type = "big"
)

## S3 method for class 'FBcases'
cr_clean(
  x,
  age_breaks = c(0, 1, seq(5, 85, 5)),
  label_tail = NULL,
  cancer_type = "big"
)

## S3 method for class 'SWcases'
cr_clean(
  x,
  age_breaks = c(0, 1, seq(5, 85, 5)),
  label_tail = NULL,
  cancer_type = "big"
)
```

```
## S3 method for class 'POP'
cr_clean(
  x,
  age_breaks = c(0, 1, seq(5, 85, 5)),
  label_tail = NULL,
  cancer_type = "big"
)
```

Arguments

x	Data of class 'FBcases', 'SWcases' or 'population'.
age_breaks	Cut points for age groups. Default is `c(0, 1, seq(5, 85, 5))`.
label_tail	Tail of the labels.
cancer_type	A character string specifying the classification method used to categorize ICD-10 codes. This determines how ICD-10 codes are classified. Options include `"big"` (classify ICD-10 codes into 26 cancer categories), `"small"` (classify ICD-10 codes into 59 cancer categories, more specific categories), `"system"` (classify ICD-10 codes into organ system), and `"gco"` (classify ICD-10 code into cancer categories same as classification published by the Global Cancer Observatory). This parameter is only available when the input data is a vector of ICD-10 codes, or object with class of `'canreg'` or `'canregs'`.

Value

Class 'canreg'.

Examples

```
data("canregs")
data <- cr_clean(canregs)

data <- cr_clean(canregs, cancer_type = "small")

data <- cr_clean(canregs[[1]], cancer_type = "big")

fbcases <- purrr::pluck(canregs[[1]], "FBcases")
fbcases <- cr_clean(fbcases)

swcases <- purrr::pluck(canregs[[1]], "SWcases")
swcases <- cr_clean(swcases)

pop <- purrr::pluck(canregs[[1]], "POP")
pop <- cr_clean(pop)
```

---

cr_filter *Filter cases from objects of class* canreg *or* canregs

---

Description

Filter cases from objects of class canreg or canregs

Usage

```
cr_filter(.data, ..., drop = c("none"), part = "all")

## S3 method for class 'canregs'
cr_filter(.data, ..., part = "all")

## S3 method for class 'canreg'
cr_filter(.data, ..., part = "all")

## S3 method for class 'asrs'
cr_filter(.data, ..., drop = c("none"))

## S3 method for class 'age_rates'
cr_filter(.data, ..., drop = c("none"))

## S3 method for class 'qualities'
cr_filter(.data, ..., drop = c("none"))

## Default S3 method:
cr_filter(.data, ..., drop = c("none"))
```

Arguments

.data	An object with class of canreg, or canregs.
...	Filtering conditions passed to dplyr::filter(). These
drop	Drop specific cancer categories.
part	A character vector specifying which components of .data to filter. Must be one or more of "FBcases", "SWcases", or "POP". Defaults to "FBcases". conditions are applied to the selected components.

Value

An object of the same class as .data (canreg or canregs) with the specified components filtered accordingly. The structure and component order are preserved.

---

cr_merge *Merge elements from objects of class* canregs, fbswicds, *or* asrs

---

Description

Merge elements from object with class of canregs, fbswicds, asrs, qualities, age_rates, or summaries into object with class of canreg, fbswicd, asr, quality, age_rate, or summary.

Usage

```
cr_merge(data)

## S3 method for class 'canregs'
cr_merge(data)
```

```
## S3 method for class 'fbswicds'
cr_merge(data)

## S3 method for class 'asrs'
cr_merge(data)

## S3 method for class 'qualities'
cr_merge(data)

## S3 method for class 'age_rates'
cr_merge(data)

## S3 method for class 'summaries'
cr_merge(data)
```

Arguments

data	An object with class of canregs, fbswicds, asrs, qualities, age_rates, or summaries.

Value

An object with merged elements.

Examples

```
data("canregs")
canreg <- cr_merge(canregs)
class(canreg)

# Merge obejct with class of `fbswicds` into obejct with class of `fbswicd`
fbsws <- count_canreg(canregs)
fbsw <- cr_merge(fbsws)

# Merge obejct with class of `asrs` into object with class of `asr`
asrs <- create_asr(canregs, year, sex, cancer, collapse = FALSE)
asr <- cr_merge(asrs)

# Merge obejct with class of `qualities` into object with class of `quality`
quas <- create_quality(canregs, year, sex, cancer, collapse = FALSE)
qua <- cr_merge(quas)

# Merge obejct with class of `age_rates` into object with class of `age_rate`
agerates <- create_age_rate(canregs, year, sex, cancer, collapse = FALSE)
agerate <- cr_merge(agerates)

# Merge obejct with class of `summaries` into object with class of `summary`
summs <- summary(canregs, collapse = FALSE)
summ <- cr_merge(summs)
```

cr_reframe *Reframe data of class canregs or fbswicds*

Description

Reframe data of class canregs or fbswicds

Usage

```
cr_reframe(x, strat = "registry")

## S3 method for class 'canregs'
cr_reframe(x, strat = "registry")

## S3 method for class 'fbswicds'
cr_reframe(x, strat = "registry")
```

Arguments

x	Object with class of canregs or fbswicds.
strat	Stratification variables used to reframe 'canregs' or 'fbswicds'.

Value

Reframed canregs or fbswicds.

Examples

```
# list reframe vars that could be used in `strat` parameter
ls_vars("reframe")
data("canregs")
# Reframe the `canregs` data according to `area_type` attribute
city <- cr_reframe(canregs, strat = "area_type")

# Reframe object with class of `canregs`
# Reframe the `canregs` according to the `province` attribute
province <- cr_reframe(canregs, strat = "province")
class(province)
names(province)

# Reframe object with class of `fbswicds`
# Convert object with class of `canregs` into object with class of `fbswicds`
fbsw <- count_canreg(canregs, cancer_type = "small")
# Reframe the `fbswicds` according to the `city` attribute
city <- cr_reframe(fbsw, strat = "city")
```

cr_select *Select elements from objects with class* "canregs"*, or* "fbswicds"

Description

This function allows you to select specific elements from objects of class 'canregs', 'fbswicds', or 'asrs' based on provided indices, logical conditions, or expressions. The selected elements are returned while preserving the class of the input object.

Usage

```
cr_select(data, ..., index = names(data))

## S3 method for class 'canregs'
cr_select(data, ..., index = names(data))

## S3 method for class 'asrs'
cr_select(data, ..., index = names(data))

## S3 method for class 'age_rates'
cr_select(data, ..., index = names(data))

## S3 method for class 'fbswicds'
cr_select(data, ..., index = names(data))

## S3 method for class 'summaries'
cr_select(data, ..., index = names(data))
```

Arguments

data	An object of class 'canregs', 'fbswicds', or 'asrs' from which elements will be selected.
...	Optional conditions or expressions used to filter elements within the list or data frame. Conditions are evaluated for each element of the input object.
index	A vector of indices specifying the elements to select. This can be a character vector (matching element names), a numeric vector (specifying positions), or a logical vector (indicating inclusion).

Value

An object of the same class as the input object, containing only the selected elements that meet the specified indices or conditions.

Examples

```
data("canregs")
# Select elements which mi greather than 0.5 from `canregs`
canregs_mi <- cr_select(canregs, mi > 0.5)

# Select elements from obejct with class of `fbswicds`
fbsws <- count_canreg(canregs)
```

```
# Select elements which `inci` greater than 250 per 100000 population
fbsws_inci <- cr_select(fbsws, inci > 250)

# Select elements from object with class of `summaries`
summ <- summary(canregs, collapse = FALSE)
# Select elements for whcih `mi` greater than 0.5
summ_mi <- cr_select(summ, mi > 0.5)
names(summ_mi)
```

`cr_write` *Clean canreg data.*

Description

Clean canreg data.

Usage

```
cr_write(x)

## S3 method for class 'canregs'
cr_write(x)

## S3 method for class 'canreg'
cr_write(x)
```

Arguments

`x`	Object with class of `canreg` or `canregs`.

Value

Object with class of `canreg` or `canregs`.

`cumrate` *Calculate the cumulative incidence or mortality rate*

Description

Computes the cumulative rate up to a specified age limit, typically used in cancer epidemiology to estimate the probability of developing or dying from a disease over a lifetime or up to a target age.

Usage

```
cumrate(
  count,
  pop,
  rate = NULL,
  eage = 70,
  agewidth = 5,
  sep_zero = TRUE,
  mp = 1,
  decimal = 6
)
```

Arguments

count	Numeric vector, number of incident cases or deaths in each age group.
pop	Numeric vector, corresponding population at risk for each age group.
rate	Numeric vector, age-specific incidence or mortality rates. If not supplied, it will be calculated as count / pop.
eage	Integer, the upper age limit (e.g., 70) up to which the cumulative rate is calculated.
agewidth	Integer, width of the age intervals (e.g., 5 for 5-year bands).
sep_zero	Logical, whether the 0–1 age group is separated (i.e., age groups are 0, 1–4, 5–9, ...). Default is TRUE.
mp	Numeric. A multiplier used to scale the final cumulative rate (e.g., 100,000 or 1). Default is 1.
decimal	Integer, number of decimal places to round the result. Default is 6.

Value

A named numeric value representing the cumulative rate, scaled by mp.

Examples

```
px <- c(
  20005, 86920, 102502, 151494, 182932, 203107, 240289, 247076, 199665,
  163820, 145382, 86789, 69368, 51207, 39112, 20509, 12301, 6586, 1909
)
dx <- c(
  156, 58, 47, 49, 48, 68, 120, 162, 160, 294, 417, 522, 546, 628,
  891, 831, 926, 731, 269
)
mx <- dx / px
cumrate(mx, eage = 70)
```

`cumrisk` *Calculate the cumulative risk*

Description

Converts a cumulative rate to a cumulative risk using the standard exponential formula. This is commonly used in cancer epidemiology to estimate the probability of developing or dying from cancer up to a certain age, under the assumption of constant rates.

Usage

```
cumrisk(cumrate, mp = 100, decimal = 2)
```

Arguments

`cumrate`	Numeric. The cumulative incidence or mortality rate, typically calculated using `cumrate`.
`mp`	Numeric. The rate multiplier used in `cumrate`. This is used for labeling purposes only. Default is `100`.
`decimal`	Integer. Number of decimal places to round the result. Default is `2`.

Details

The cumulative risk is calculated as:

$$1 - \exp(-\text{cumrate})$$

This converts the cumulative rate to a probability, assuming the event rate is constant over each age interval and the competing risks are ignored.

Value

A named numeric value representing the cumulative risk (as a percentage).

Examples

```
px <- c(
  20005, 86920, 102502, 151494, 182932, 203107, 240289, 247076, 199665,
  163820, 145382, 86789, 69368, 51207, 39112, 20509, 12301, 6586, 1909
)
dx <- c(
  156, 58, 47, 49, 48, 68, 120, 162, 160, 294, 417, 522, 546, 628,
  891, 831, 926, 731, 269
)
mx <- dx / px
cumrate(mx, eage = 70)
cumrisk(cumrate(mx, eage = 70))
```

cutage *Group ages into categories*

Description

Groups numeric age values into categorized age bands using one of three methods: fixed interval ("interval"), equal distance ("distance"), or quantile-based grouping ("quantile"). Supports flexible labeling and language-specific suffixes.

Usage

```
cutage(
  x,
  method = "distance",
  length = 5,
  maxage = 85,
  sep_zero = TRUE,
  breaks = c(seq(0, 85, 5)),
  labels = NULL,
  lang = "cn",
  label_tail = NULL,
  right = FALSE
)
```

Arguments

x	Numeric vector of ages.
method	Character. Grouping method: "interval" (custom breaks), "distance" (uniform width), or "quantile" (equal-sized groups).
length	Integer. Width of age bands (used only if method = "distance"). Default is 5.
maxage	Numeric. Upper limit of the age range (used in "distance" method).
sep_zero	Logical. Whether to separate age 0 into its own group (only used if method = "distance"). Default is TRUE.
breaks	Numeric vector of breakpoints (required if method = "interval").
labels	Character vector of labels for resulting age groups. If NULL, default interval-style labels are generated.
lang	Output language for default labels, "cn" (Chinese) or "en" (English). Default is "cn".
label_tail	Character string appended to labels, e.g., "yrs". Default depends on lang.
right	Logical. Whether intervals are right-closed. Passed to cut(). Default is FALSE.

Value

A factor variable of age groups with labeled levels.

Examples

```
ages <- sample(0:101, 200, replace = TRUE)
cutage(ages, method = "distance", length = 5, maxage = 60, sep_zero = TRUE)
# Custom breaks
cutage(ages, method = "interval", breaks = c(0, 15, 30, 45, 60, 75, Inf))
# Quantile-based grouping
cutage(ages, method = "quantile")
```

---

del_dict_files *Delete a dictionary file from canregtools config*

Description

Removes a specified dictionary file (e.g., "registry", "area_type", "custom") from the user's local configuration directory.

Usage

```
del_dict_files(dict = "custom")
```

Arguments

dict	A character string specifying the dictionary name. Default is "custom".

Value

Invisibly returns TRUE if deleted, FALSE otherwise.

Examples

```
del_dict_files("custom")
```

---

draw_barchart *Draw Grouped Bar Charts with Facets*

Description

This function creates grouped bar charts with optional faceting. It is useful for comparing category-specific values across different groups and panels.

Usage

```
draw_barchart(
  data,
  x,
  y,
  group = NULL,
  facet = NULL,
  facet_label = NULL,
  y_label = NULL,
  rev_group = FALSE,
  grid = NULL,
  topn = NULL,
  axis = NULL,
  bar_side = NULL,
  bar_way = NULL,
  gap = NULL,
  csize = 0.8,
  space = 0.9,
  adj = -0.01,
  gl = NULL,
  cols = NULL,
  palette = "Peach",
  x_label_side = 1,
  legend = FALSE,
  legend_label = NULL,
  legend_pos = c(0.4, 0.2),
  dens = c(-1, -1),
  overlay = FALSE
)
```

Arguments

data	A data.frame or tibble containing the input data.
x	The categorical variable (unquoted) for the x-axis.
y	The numerical variable (unquoted) for the y-axis (bar height).
group	Optional grouping variable (unquoted) used for color/fill aesthetics.
facet	Optional faceting variable (unquoted) for splitting the data into panels.
facet_label	Optional character vector of labels corresponding to each facet.
y_label	Optional y-axis label or vector of labels for facets.
rev_group	Reverse the bar position of group variables.
grid	A numeric vector of length 2 indicating the layout (rows, columns) of the plot grid.
topn	Number of top x categories to display in each group/panel.
axis	Optional vector of breaks for the y-axis. If NULL, computed automatically.
bar_side	Integer indicating the side of bars: 1 = left, 2 = right.
bar_way	Integer for bar layout: 1 = one-sided, 2 = mirrored.
gap	Horizontal spacing between bars and y-axis (default depends on x_label_side).
csize	Character size scaling factor (default = 0.8).

space	Vertical space between bars (default = 0.9).
adj	Vertical adjustment of x-label text (default = -0.01).
gl	Integer. Indicating the line type of the grids.
cols	Optional vector of colors for bars.
palette	Character. Name of palette colors.
x_label_side	Side to place x-labels: 1 = inside, 2 = outside (default = 1).
legend	Logical. Whether to show a legend (default = FALSE).
legend_label	Optional character vector for legend labels.
legend_pos	Optional, numeric vector of length 2 for legend position.
dens	A numeric vector of two density values for bar shading (default = c(-1, -1)).
overlay	Logical. Whether to overlay bars from two groups in the same panel.

Value

A base R plot with grouped bar charts, optionally faceted.

Examples

```
data("canregs")
asr <- create_asr(canregs[[1]], year, sex, cancer, event = "fbs")
asr <- cr_filter(asr, drop = c("total", "others"))
draw_barchart(asr, x = cancer, y = cr, group =  year, facet = sex)
```

---

draw_dumbbell *Plot dumbbell chart*

Description

Plot dumbbell chart

Usage

```
draw_dumbbell(
  data,
  x = NULL,
  y1 = NULL,
  y2 = NULL,
  topn = 20,
  sort = "insc",
  legend = NULL,
  cols = c("#006400", "gray", "#b32134"),
  gl = NULL,
  gl_col = c("gray"),
  main = ""
)
```

Arguments

data	A data frame contains data to be plotted.
x	A category variable in data.
y1	Variable indicate start point.
y2	Variable indicate end point.
topn	Top n values to be plotted.
sort	Sort options.
legend	Legends.
cols	Colors of the start and end points.
gl	Integer. Indicating the line type of the grids.
gl_col	Color of the background grid.
main	Main title of the plot.

Value

A dumbbell plot.

Examples

```
asr <- create_asr(canregs[[1]], year, cancer, show_ci = TRUE) |>
  drop_others() |>
  drop_total() |>
  add_labels(vars = "cancer", lang = "en", label_type = "abbr")
draw_dumbbell(asr, "cancer_en", asr_lower_cn2000, asr_upper_cn2000, topn = 15)
```

---

draw_linechart *Draw a Custom Line Chart Using Base R*

Description

This function draws a line chart from a data frame, optionally grouped by a categorical variable. It uses base R graphics and supports custom axis ticks, labels, and styles.

Usage

```
draw_linechart(
  data,
  x,
  y,
  group = NULL,
  facet = NULL,
  grid = c(1, 1),
  x_axis = NULL,
  y_axis = NULL,
  x_label = NULL,
  y_label = NULL,
  axis_title = c("Age (years)", "Age specific rate"),
```

```
  cols = NULL,
  palette = "Peach",
  line_type = "l",
  lwd = 2,
  adj = 0.02,
  srt = 60,
  main = NULL,
  sub = NULL,
  legend_pos = c(0.05, 0.95),
  mar = c(1, 0, 1, 0),
  add = TRUE,
  offset = 0.01,
  ...
)
```

Arguments

data	A data frame containing the variables to plot.
x, y	Bare column names for the x and y axis variables.
group	Optional bare column name used to group and color lines.
facet	Optional bare column name used for faceting. If provided, the data will be split by this variable and plotted in a multi-panel layout.
grid	A vector of length 2 specifying number of rows and columns for facets. Default is c(1, 1).
x_axis	Optional numeric vector specifying x-axis tick locations.
y_axis	Optional numeric vector specifying y-axis tick locations.
x_label, y_label	Optional labels for x and y axis ticks. If NULL, defaults are used.
axis_title	Character vector of length 2 giving the axis titles: c("x axis label", "y axis label").
cols	Character vector of line colors. Defaults to c("darkgreen", "darkred", "gray").
palette	Character, palette name indicate group of colors.
line_type	1-character string giving the type of plot desired. The following values are possible, for details, see plot: "p" for points, "l" for lines, "b" for both points and lines, "c" for empty points joined by lines, "o" for overplotted points and lines, "s" and "S" for stair steps and "h" for histogram-like vertical lines. Finally, "n" does not produce any points or lines.
lwd	Line width. Default is 2.
adj	Adjustment for axis text placement. Default is 0.02.
srt	String rotation angle for x-axis labels. Default is 90 degrees.
main, sub	Main title and subtitle of the plot.
legend_pos	Position of the legend.
mar	Margin of the sub plot.
add	Logical. If TRUE, restores original graphics parameters after plotting.
offset	Axis offset used for spacing ticks. Default is 0.01.
...	Additional arguments (currently unused).

Examples

```
data("canregs")
fbsw <- count_canreg(canregs[[1]], label_tail="yrs")
agerate <- create_age_rate(fbsw, year, sex)
agerate <- add_labels(agerate, lang = "en")
draw_linechart(agerate, agegrp, rate, sex)
agerate <- create_age_rate(fbsw, year, sex, cancer)
agerate <- add_labels(agerate, lang = "en")
agerate <- dplyr::filter(agerate, cancer %in% as.character(c(103:106)))
draw_linechart(agerate, agegrp, rate, sex, cancer, grid = c(2, 2))
```

---

draw_pyramid *Plot a population pyramid*

Description

This function draws a population pyramid using either raw population numbers or proportions, displaying age groups in the center and population counts (e.g., by sex) on each side.

Usage

```
draw_pyramid(
  data,
  x,
  y,
  group,
  facet = NULL,
  facet_label = NULL,
  grid = NULL,
  show_value = FALSE,
  show_prop = TRUE,
  left_axis = NULL,
  right_axis = NULL,
  left_label = NULL,
  right_label = NULL,
  cgap = 0.3,
  cstep = 1,
  csize = 1,
  labs = c("Males", "Ages", "Females"),
  gl = 2,
  cadj = 0,
  cols = c("#006400", "#b32134"),
  dens = c(-1, -1),
  main = "",
  ...
)
```

Arguments

data	A data.frame containing the variables for age group (x), population (y), and grouping variable (group, e.g., sex). It must contain at least three columns: age group (x-axis labels), population size, and grouping (e.g., "male" and "female").

x	A variable indicating age groups (quoted or unquoted).
y	A variable indicating population counts (quoted or unquoted).
group	A grouping variable, typically representing sex (quoted or unquoted).
facet	Optional unquoted variable to facet the data by (e.g., year, region). A separate pyramid will be drawn for each unique value.
facet_label	Optional character vector of labels for each facet. If NULL, labels will be extracted from the unique values of facet.
grid	Optional vector of two integers specifying the layout of the facet grid (number of rows, number of columns). If NULL, it defaults to a vertical stack (c(n, 1) for n facets).
show_value	Logical. If TRUE, displays the actual population values beside the bars. Default is FALSE.
show_prop	Logical. If TRUE, the bars represent proportions (%) rather than absolute values. Default is TRUE.
left_axis	Numeric vector of tick marks for the left side (e.g., males). If NULL, it will be generated using pretty().
right_axis	Numeric vector of tick marks for the right side (e.g., females). If NULL, it will use left_axis.
left_label	Character vector to customize axis labels on the left side. If NULL, generated using formatC().
right_label	Character vector for axis labels on the right side. Same rules as left_label.
cgap	Numeric. Width of the central gap (relative to axis length). Default is 0.3.
cstep	Integer. Step interval between age group labels. Default is 1 (every label shown).
csize	Numeric. Scaling factor for text and lines. Default is 1.
labs	A character vector of three labels: left side (e.g., "Males"), center (e.g., "Ages"), and right side (e.g., "Females"). Default is c("Males", "Ages", "Females").
gl	Integer. Indicating the line type of the grids.
cadj	Numeric. Vertical adjustment for center age labels. Default is 0.
cols	A character vector of two colors for the left and right bars. Default is c("#006400", "#b32134").
dens	A numeric vector indicating shading densities (lines per inch) for bars. Use -1 to fill solid bars. Default is c(-1, -1).
main	A character string for the main plot title. Default is an empty string.
...	Additional graphical parameters passed to the base plot() function.

Value

A base R graphics pyramid plot. It does not return a value.

Examples

```
data("canregs")
pop <- canregs[[1]]$POP
draw_pyramid(pop, agegrp, rks, sex)
```

drop_others *Drop "other" cancer codes*

Description

Filters out cases classified as "other" cancer types based on specific codes.

Usage

```
drop_others(x)
```

Arguments

x	A data frame containing a cancer variable created by classify_icd10().

Value

A filtered data frame excluding non-specific cancer codes.

Examples

```
data("canregs")
asr <- create_quality(canregs, year, sex, cancer)
asr2 <- asr |> drop_others()
```

drop_total *Drop the total cancer*

Description

Filter out cases that summarized as total when using create_asr(), create_quality(), or create_age_rate().

Usage

```
drop_total(x)
```

Arguments

x	A data frame containing a cancer variable created by classify_icd10().

Value

A filtered data frame excluding cancer in c("60", "61").

Examples

```
data("canregs")
qua <- create_quality(canregs, year, sex, cancer)
qua2 <- qua |> drop_total()
```

esti_fbswicd *Estimate object of fbswicd*

Description

This function estimates `fbswicd` object based on existing `fbswicd` object and using the total population data.

Usage

```
esti_fbswicd(obj, pop = NULL)
```

Arguments

`obj`	An object with class of `fbswicd`.
`pop`	An population dataset to override the population in `obj`.

Details

The estimation proceeds in the following steps:

1. Calculate age-specific incidence and mortality rates for different sex and cancer sites.
2. Derive estimated counts of `fbs` and `sws` using the rate × population formula.
3. Recalculate proportions (e.g., `mv`, `dco`) back to estimated case counts.
4. Join all information and return a new object of the same class as `obj`.

Value

An estimated object with class of `fbswicd`.

See Also

`create_age_rate()`, `cr_filter()`

esti_pop *Estimating population structure using interpolation method.*

Description

Estimating population structure using interpolation method.

Usage

```
esti_pop(pop1, pop2, period)
```

Arguments

`pop1`	Population or population proportion in each age group for the start year.
`pop2`	Population or population proportion in each age group for the end year.
`period`	Vector contain the start year and end year value.

Value

A data frame contain the estimated population proportion in each year during the period with each year in one column and each age group in one row.

Examples

```
pop1 <- c(
  59546, 294129, 472511, 552549, 821119, 996436, 805635, 1004506,
  989357, 1056612, 986559, 792270, 544544, 452297, 473579, 350802,
  212614, 109598, 61990
)
pop2 <- c(
  75641, 377276, 327116, 380338, 539034, 1158852, 1152329, 881443,
  903484, 1011164, 1238871, 1137832, 1022787, 645441, 464777,
  482941, 406144, 227977, 144526
)
esti_pop(pop1, pop2, c(2000, 2010))
```

---

expand_age_pop *Expand population data from 5-year age groups to single-year ages*

Description

`expand_age_pop` transforms population data aggregated in age groups into estimates for single-year ages. It utilizes interpolation methods to distribute the grouped data across individual ages, ensuring consistency with the original totals.

Usage

```
expand_age_pop(x, method = "linear")
```

Arguments

`x`	A numeric vector representing the population counts for each age group. The vector should have 19 elements corresponding to the following age groups: 0, 1–4, 5–9, ..., 85+.
`method`	A character string specifying the interpolation method to use. Options include: • `"linear"`: Linear interpolation. • `"constant"`: Constant interpolation. • `"periodic"`: Periodic spline interpolation. • `"natural"`: Natural spline interpolation. The default is `"linear"`.

Value

A data frame with two columns:

`x` Integer ages from 0 to 92.

`y` Estimated population counts for each single-year age.

Examples

```
# Example population data for 19 age groups: 0, 1-4, 5-9, ..., 85+
ages <- c(
  5053, 17743, 25541, 32509, 30530, 34806, 36846, 38691, 40056,
  39252, 37349, 30507, 26363, 21684, 15362, 11725, 7461, 3260, 915
)
eages <- expand_age_pop(ages)
head(eages)
```

---

expand_lifetable *Expand an five-year abridged life table to complete life table*

Description

`expand_lifetable()` transforms a five-year abridged life table into a one-year complete life table using the Elandt–Johnson method.

Usage

```
expand_lifetable(lx)
```

Arguments

`lx`	A numeric vector representing the number of survivors (l_x) at the beginning of each age interval in the abridged life table. The vector should correspond to age intervals: 0, 1–4, 5–9, ..., up to the oldest age group.

Value

A list containing:

fitlx A numeric vector of length 101 representing the estimated number of survivors at each single year of age from 0 to 100.

fitmx A numeric vector of length 101 representing the estimated central death rates (m_x) for each single year of age from 0 to 100.

References

Baili, P., Micheli, A., Montanari, A., & Capocaccia, R. (2005). Comparison of Four Methods for Estimating Complete Life Tables from Abridged Life Tables Using Mortality Data Supplied to EUROCARE-3. *Mathematical Population Studies*, 12(4), 183–198. https://doi.org/10.1080/08898480500301751

Examples

```
# Example abridged life table data (normalized to a radix of 1)
lx <- c(
  100000, 99498.39, 99294.62, 99173.88, 99047.59, 98840.46,
  98521.16, 98161.25, 97636.99, 96900.13, 95718.96, 93930.91,
  91463.21, 87131.41, 80525.02, 70907.59, 58090.75, 41630.48,
  24019.33
)
lx <- lx / 100000
expand_lifetable(lx)
```

get_pop *Subset or summarize population data*

Description

Extracts population data from canreg-style objects and optionally summarizes it by specified grouping variables.

Usage

```
get_pop(data, sum_by = NULL, collapse = FALSE)

## S3 method for class 'canreg'
get_pop(data, sum_by = NULL, collapse = FALSE)

## S3 method for class 'canregs'
get_pop(data, sum_by = NULL, collapse = FALSE)

## S3 method for class 'fbswicd'
get_pop(data, sum_by = NULL, collapse = FALSE)

## S3 method for class 'fbswicds'
get_pop(data, sum_by = NULL, collapse = FALSE)
```

Arguments

data	An object of class canreg, canregs, fbswicd, or fbswicds.
sum_by	Character vector of grouping variables to summarize population.
collapse	Logical, if TRUE, collapses the list of results into a single data frame (only for list-type objects).

Value

A data frame or a list of data frames depending on collapse.

get_stdpop *Get standardized population data*

Description

Retrieves standardized population data for a specified standard from dict_maps.

Usage

```
get_stdpop(std = "wld85", sep_zero = TRUE)
```

Arguments

std	Character string specifying the population standard. Supported values are "cn64", "cn82", "cn2000", "wld85", "wld2000". Defaults to "wld85".
sep_zero	Logical value indicating whether age 0 should be treated as a separate group.

Value

A vector or data structure containing the standardized population data for the specified standard, or NULL if the standard is not supported.

Examples

```
## Not run:
  get_std("cn64")
  get_std("wld2000")

## End(Not run)
```

---

ls_attrs *list built-in attributes name used in* cr_reframe()

Description

list built-in attributes name used in cr_reframe()

Usage

```
ls_attrs()
```

Value

A character vector.

Examples

```
ls_attrs()
```

---

ls_dict *list dictionary used in package*

Description

list dictionary used in package

Usage

```
ls_dict(dict = "registry")
```

Arguments

dict	Character, name of dictionary.

Value

A tibble of dictionary.

Examples

```
ls_dict("registry")
ls_dict("area_type")
```

---

ls_dict_files *List dictionary files used by canregtools*

Description

Lists all dictionary `.rds` files stored in the user's config directory for `canregtools`.

Usage

```
ls_dict_files(full.names = FALSE, with_info = FALSE)
```

Arguments

full.names	Logical. Whether to return full file paths. Default is `FALSE`.
with_info	Logical. Whether to return file size and last modified time. Default is `FALSE`.

Value

A character vector of file names, or a tibble with file info if `with_info = TRUE`.

Examples

```
ls_dict_files()
ls_dict_files(full.names = TRUE)
ls_dict_files(with_info = TRUE)
```

ls_vars *List variable names and their descriptions by category*

Description

This function returns a `tibble` containing variable names and their descriptions based on the specified category.

Usage

```
ls_vars(type = "std")
```

Arguments

`type`	A character string specifying the type of variables to list. Options are `"std"` for standard population variables, `"summary"` for summary variables, and `"reframe"` for variables used in `cr_reframe()`.

Value

A tibble with five columns:

- `code`: The name of the variable.
- `cname`: A detailed description of the variable in Chinese.
- `ename`: A detailed description of the variable in English.
- `abbr_cn`: An abbreviated description of the code label in Chinese.
- `abbr_en`: An abbreviated description of the code label in English.

See Also

`cr_reframe()`, `summary()`, `create_asr()`

Examples

```
ls_vars("std")
ls_vars("summary")
ls_vars("reframe")
```

lt *Compute a life table from age-specific mortality rates*

Description

`lt()` constructs a life table based on a vector of age-specific mortality rates (mx), starting age, age group width, and specified sex. It calculates standard life table columns including the probability of dying (qx), number of survivors (lx), number of deaths (dx), person-years lived (Lx), total person-years remaining (Tx), and life expectancy (ex).

Usage

```
lt(mx, sage = 0, sep_zero = TRUE, age_width = 5, sex = "male")
```

Arguments

mx	Numeric vector of age-specific mortality rates.
sage	Integer. Starting age of the first age group (default is 0).
sep_zero	Logical. Indicates whether the age 0 group is separated from the 1–4 age group (default is TRUE).
age_width	Integer. Width of each age group in years; typically 1 or 5 (default is 5).
sex	Character string specifying the sex: "male", "female", or "total" (default is "male").

Details

The function uses standard demographic formulas to compute life table values. The average number of person-years lived in the interval by those dying in the interval (ax) is estimated using the `calc_ax()` function, which applies formulas based on the specified sex and age group width. The calculations assume a radix (starting population) of 1.

Value

A data frame with the following columns:

age Starting age of each age group.

mx Age-specific mortality rate.

qx Probability of dying between age x and x+n.

lx Number of survivors at exact age x, starting from a radix of 1.

dx Number of deaths between ages x and x+n.

Lx Person-years lived between ages x and x+n.

Tx Total person-years remaining after age x.

ex Life expectancy at exact age x.

Examples

```
# Example 1: Using mortality rates derived from death and population counts
px <- c(
  20005, 86920, 102502, 151494, 182932, 203107, 240289, 247076, 199665,
  163820, 145382, 86789, 69368, 51207, 39112, 20509, 12301, 6586, 1909
)
dx <- c(
  156, 58, 47, 49, 48, 68, 120, 162, 160, 294, 417, 522, 546, 628,
  891, 831, 926, 731, 269
)
mx <- dx / px
lt(mx, sage = 0, age_width = 5, sex = "male")

# Example 2: Using predefined mortality rates
mx <- c(
  0.01685, 0.00085, 0.00044, 0.00045, 0.00064, 0.00086, 0.00103,
  0.00136, 0.00195, 0.00291, 0.00429, 0.00672, 0.00985, 0.01596,
```

```
  0.02605, 0.04536, 0.07247, 0.12078, 0.17957, 0.25938, 0.25989
)
lt(mx, sage = 0, age_width = 5, sex = "total")
```

`lt2` *Compute a life table from age-specific mortality rates*

Description

`lt()` constructs a life table based on a vector of age-specific mortality rates (mx), starting age, age group width, and specified sex. It calculates standard life table columns including the probability of dying (qx), number of survivors (lx), number of deaths (dx), person-years lived (Lx), total person-years remaining (Tx), and life expectancy (ex).

Usage

```
lt2(
  Dx = NULL,
  Cdx = NULL,
  Px = NULL,
  mx = NULL,
  sage = 0,
  sep_zero = TRUE,
  age_width = 5,
  sex = "male",
  cohort = 1e+05
)
```

Arguments

`Dx`	description
`Cdx`	description
`Px`	Average population size.
`mx`	Numeric vector of age-specific mortality rates.
`sage`	Integer. Starting age of the first age group (default is 0).
`sep_zero`	Logical. Indicates whether the age 0 group is separated from the 1–4 age group (default is TRUE).
`age_width`	Integer. Width of each age group in years; typically 1 or 5 (default is 5).
`sex`	Character string specifying the sex: "male", "female", or "total" (default is "male").
`cohort`	The size of the initial cohort.

Details

The function uses standard demographic formulas to compute life table values. The average number of person-years lived in the interval by those dying in the interval (ax) is estimated using the `calc_ax()` function, which applies formulas based on the specified sex and age group width. The calculations assume a radix (starting population) of 1.

Value

A data frame with the following columns:

age Starting age of each age group.

mx Age-specific mortality rate.

qx Probability of dying between age x and x+n.

lx Number of survivors at exact age x, starting from a radix of 1.

dx Number of deaths between ages x and x+n.

Lx Person-years lived between ages x and x+n.

Tx Total person-years remaining after age x.

ex Life expectancy at exact age x.

Examples

```
# Example 1: Using mortality rates derived from death and population counts
px <- c(
  20005, 86920, 102502, 151494, 182932, 203107, 240289, 247076, 199665,
  163820, 145382, 86789, 69368, 51207, 39112, 20509, 12301, 6586, 1909
)
dx <- c(
  156, 58, 47, 49, 48, 68, 120, 162, 160, 294, 417, 522, 546, 628,
  891, 831, 926, 731, 269
)
mx <- dx / px
lt(mx, sage = 0, age_width = 5, sex = "male")

# Example 2: Using predefined mortality rates
mx <- c(
  0.01685, 0.00085, 0.00044, 0.00045, 0.00064, 0.00086, 0.00103,
  0.00136, 0.00195, 0.00291, 0.00429, 0.00672, 0.00985, 0.01596,
  0.02605, 0.04536, 0.07247, 0.12078, 0.17957, 0.25938, 0.25989
)
lt(mx, sage = 0, age_width = 5, sex = "total")
```

---

quality *Quality Indicators from the China Cancer Registry Annual Report*

Description

This dataset contains key quality indicators of population-based cancer registry (PBCR) data, as published in the China Cancer Registry Annual Report by the National Cancer Center of China.

Usage

```
quality
```

Format

A data frame with quality indicators for various cancer types across different years and area types.

`year` Calendar year of the PBCR data.

`area_type` Area classification code: `910000` for urban areas, `920000` for rural areas.

`cancer` Cancer type code.

`mv` Proportion of morphologically verified (MV) cases.

`dco` Proportion of cases identified through sdeath certificates only (DCO).

`mi` Mortality-to-incidence (MI) ratio.

Source

China Cancer Registry Annual Report, National Cancer Center, China.

Examples

```
data("quality")
```

---

read_canreg *Read cancer registration data*

Description

Reads cancer registry data from one or more Excel files or from a directory containing Excel files. It extracts case and population data and returns objects of class `"canreg"` or a list of such objects (`"canregs"`).

Usage

```
read_canreg(
  x,
  pop_type = "long",
  age_var = "agegroup",
  pop_var = "popu",
  death_var = "death"
)
```

Arguments

`x`	A path to an Excel file, a character vector of file paths, or a directory containing Excel files.
`pop_type`	A character string specifying the format of the population sheet. Must be either `"long"` (default) or `"wide"`.
`age_var`	Name of the age group column (used only when `pop_type = "long"`). Default is `"agegroup"`.
`pop_var`	Name of the population count column (used only when `pop_type = "long"`). Default is `"popu"`.
`death_var`	Name of the death count column (used only when `pop_type = "long"`). If not present in the sheet, a column with 0s will be added. Default is `"death"`.

Value

A canreg object (a list with components: areacode, FBcases, SWcases, POP) if a single file is read. A named list of such objects with class "canregs" if multiple files or a directory is provided.

Examples

```
## Not run:
file_address <- "410302.xlsx"
canreg <- read_canreg(file_address, pop_type = "long")

## End(Not run)
```

---

show_registry *Get the areacodes of the registry*

Description

Query the area codes corresponding to a given registry type.

Usage

```
show_registry(regi_type = 1:4)
```

Arguments

regi_type	Numeric or character vector indicating registry types. Defaults to 1:4.

Value

A character vector of area codes.

Examples

```
show_registry(1:4)
show_registry(c("1", "2"))
```

---

summary.canreg *Summary object of class 'canreg'*

Description

Summary object of class 'canreg'

Usage

```
## S3 method for class 'canreg'
summary(object, collapse = FALSE, ...)

## S3 method for class 'canregs'
summary(object, collapse = TRUE, ...)
```

Arguments

object	Object data with class of 'canreg', 'canregs'
collapse	Collapse data or not.
...	Other filter expressions

Value

A data frame contains summary statistics of canreg data.

Examples

```
data("canregs")
data <- canregs[[1]]
summary(data)


summary(canregs)
```

---

tidy_age *Tidy age description*

Description

Parses age descriptions written in Chinese and converts them into numeric values expressed in years, months, or days. It interprets age strings containing Chinese characters such as (years), (months), and (days), and converts them to a numeric vector representing age in the specified unit.

Usage

```
tidy_age(x, unit = "year")
```

Arguments

x	A character vector containing age descriptions in Chinese.
unit	A character string specifying the unit of the returned values. Options are "year" (default), "month", or "day".

Value

A numeric vector representing ages in the specified unit:

- year: Truncated age in years.
- month: Truncated age in months.
- day: Rounded age in days.

Examples

```
agedes <- c(
  "50\u5c8110\u67083\u6708", "19\u5c815\u6708",
  "1\u5c8130\u6708", "3\u670820\u6708", "30\u6708"
)
tidy_age(agedes, unit = "year")
tidy_age(agedes, unit = "month")
tidy_age(agedes, unit = "day")
```

`tidy_sex` *Tidy gender variable*

Description

Standardizes gender-related values into consistent numeric codes or factors. This function maps various gender-related character strings (e.g., "male", "female", "man", "woman", "1", "2", etc.) to standardized numeric values: `1` for male, `2` for female, and `0` for total. It supports both Chinese and English labels. Optionally, the result can be returned as a factor with appropriate labels.

Usage

```
tidy_sex(x, lang = "cn", as_factor = FALSE)
```

Arguments

`x`	A character or numeric vector containing gender information.
`lang`	Character, specify the output language, options are 'cn', or 'en', default is 'cn'.
`as_factor`	Logical, indicate whether output value as factor.

Value

A numeric vector or a factor representing gender:

0 Total

1 Male

2 Female

If `as_factor = TRUE`, a factor is returned with labels in the specified language (`lang`).

Examples

```
gender <- c("male", "men", "women", "female", "women", "man", "1", "2")
tidy_sex(gender)
```

`tidy_var` *Reformat variable values for Cancer Registration in China*

Description

Standardizes and labels values of a specified variable according to the national cancer registration standard of China: T/CHIA 18-2021.

Usage

```
tidy_var(
  x,
  var_name = "occu",
  label_type = "full",
  lang = "code",
  sep = "",
  as_factor = FALSE
)
```

Arguments

`x`	A character vector containing raw values of a variable used in cancer registry data.
`var_name`	A character string indicating the name of the variable to reformat (e.g., `"occu"` for occupation). Must be one of the variable names defined in `tidy_var_maps`.
`label_type`	Type of the label used ("full" or "abbr").
`lang`	Character, specify the output language, options are 'cn', or 'en', default is 'cn'.
`sep`	A character string to separate the label.
`as_factor`	Logical, indicate whether output value as factor.

Details

`tidy_var()` converts raw character inputs into standardized labels, codes, or abbreviations based on reference mappings defined for each variable (e.g., occupation, basis of diagnosis, etc.). It supports both Chinese and English outputs and can return values as factors with labeled levels.

Value

A character or factor vector of reformatted values. The output depends on the settings for `label_type`, `lang`, and `as_factor`:

- If `as_factor = FALSE`, returns a character vector.
- If `as_factor = TRUE`, returns a factor with sorted unique levels.
- The labels used depend on `lang` (`"cn"`, `"en"`, `"code"`, or `"icd10"`) and `label_type` (`"full"` or `"abbr"`).

Examples

```
occu <- c("11", "13", "17", "21", "24", "27", "31", "37", "51", "80", "90")
tidy_var(occu, var_name = "occu", lang = "cn")
tidy_var(occu, var_name = "occu", lang = "en")
tidy_var(occu, var_name = "occu", lang = "cn", label_type = "abbr")
tidy_var(occu, var_name = "occu", lang = "en", label_type = "abbr")
```

---

truncrate *Calculate truncated age standardized rate*

Description

Calculates the **truncated age-standardized rate** (ASR) over a specified age range (e.g., 35–64 years). It uses the **direct method of standardization** by applying age-specific rates to a standard population within the truncated age group. This is particularly useful when comparing disease rates in middle-aged or other focused subgroups.

Usage

```
truncrate(
  cases,
  pop,
  stdpop = NULL,
  trunc_age = c(35, 64),
  agewidth = 5,
  sep_zero = TRUE,
  mp = 100,
  decimal = 2
)
```

Arguments

cases	Number of cases.
pop	Number of population at risk.
stdpop	The standard population.
trunc_age	The truncated age range.
agewidth	Age groups width, default is 5.
sep_zero	Logical value, if the 0 age group was a separate group.
mp	A multiplier used to scale the calculated rates. Default is 100.
decimal	Decimals of the calculated rates, default is 2.

Value

Truncated age standardized rate.

Examples

```
px <- c(
  20005, 86920, 102502, 151494, 182932, 203107, 240289, 247076, 199665,
  163820, 145382, 86789, 69368, 51207, 39112, 20509, 12301, 6586, 1909
)
dx <- c(
  156, 58, 47, 49, 48, 68, 120, 162, 160, 294, 417, 522, 546, 628,
  891, 831, 926, 731, 269
)
stdpop <- c(2.4, 9.6, 10, 9, 9, 8, 8, 6, 6, 6, 6, 5, 4, 4, 3, 2, 1, 0.5, 0.5)
truncrate(dx, px, stdpop, trunc_age = c(35, 64))
```

`write_areacode` *Write custom six-digit administrative division codes to dictionary*

Description

Saves user-defined administrative division codes and their associated labels (in both Chinese and English) to the local dictionary used by `canregtools`.

Usage

```
write_areacode(x = NULL, cache_refresh = FALSE)
```

Arguments

`x`	A data frame containing at least the following columns: `areacode`, `cname`, `ename`, `abbr_cn`, and `abbr_en`.
`cache_refresh`	Logical. If TRUE, refresh the dictionary to default values before updating.

Value

Invisibly returns `NULL`. The function is called for its side effect.

Examples

```
## Not run:
dict <- data.frame(
  areacode = c("410302"),
  cname = c("\u8001\u57CE\u533A"),
  ename = c("Laocheng District"),
  abbr_cn = c("\u8001\u57CE"),
  abbr_en = c("Laocheng")
)
write_areacode(dict)

## End(Not run)
```

write_registry *Write registry attributes to a user-defined dictionary*

Description

Stores the mapping between six-digit administrative division codes areacode and their corresponding attributes e.g., registry names or area types into a dictionary file. This supports consistent labeling and downstream processing in cancer registry data.

Usage

```
write_registry(
  x = NULL,
  dict = "registry",
  cache_refresh = FALSE,
  quiet = TRUE
)

## S3 method for class 'data.frame'
write_registry(
  x = NULL,
  dict = "registry",
  cache_refresh = FALSE,
  quiet = TRUE
)

## S3 method for class 'list'
write_registry(
  x = NULL,
  dict = "registry",
  cache_refresh = FALSE,
  quiet = TRUE
)

## S3 method for class '`NULL`'
write_registry(
  x = NULL,
  dict = "registry",
  cache_refresh = FALSE,
  quiet = TRUE
)

## S3 method for class 'character'
write_registry(
  x = NULL,
  dict = "registry",
  cache_refresh = FALSE,
  quiet = TRUE
)
```

Arguments

x	A named list, a named character vector or a data frame that includes the columns areacode and the target attribute e.g., registry, area_type.
dict	A character string specifying the type of dictionary to update. Supported values are "registry" and "area_type".
cache_refresh	If TRUE, refresh the dictionary to default values before updating.
quiet	If TRUE, message will be supressed.

Details

This function allows users to build or update local mapping dictionaries for area-level attributes. It supports multiple input formats and updates internal files saved in the user-specific R cache directory.

Value

A tibble with two columns: areacode and the corresponding attribute values.

Examples

```
write_registry(list('410302' = '410301'))

# Registry attributes stored in data frame
registry_dict <- data.frame(
areacode = c("410302", "410303", "410304", "410305", "410306", "410307"),
registry = c(rep("410301", 5), "410300"),
area_type = rep("urban", 6)
)
write_registry(registry_dict)

# Registry attributes stored in list
dict <- list(
'410302' = '410301',
'410303' = '410301',
'410304' = '410301',
'410305' = '410301',
'410306' = '410301',
'410307' = '410301'
)
write_registry(dict)

# Registry attributes using built-in information
dict <- NULL
write_registry(dict)

# Registry attributes stored in named character vector with areacode as
# name and attributes as values
dict <- rep("410301", 5)
names(dict) <- c("410302", "410303", "410304", "410305", "410306")
write_registry(dict)
```

Index