Package 'PHEindicatormethods'

May 7, 2026

Type Package

Version 2.1.1

Title Common Public Health Statistics and their Confidence Intervals

Description Functions to calculate commonly used public health statistics and their confidence intervals using methods approved for use in the production of Public Health England indicators such as those presented via Fingertips (<https://fingertips.phe.org.uk/>). It provides functions for the generation of proportions, crude rates, means, directly standardised rates, indirectly standardised rates, standardised mortality ratios, slope and relative index of inequality and life expectancy.
Statistical methods are referenced in the following publications.
Breslow NE, Day NE (1987) <doi:10.1002/sim.4780080614>.
Dobson et al (1991) <doi:10.1002/sim.4780100317>.
Armitage P, Berry G (2002) <doi:10.1002/9780470773666>.
Wilson EB. (1927) <doi:10.1080/01621459.1927.10502953>.
Altman DG et al (2000, ISBN: 978-0-727-91375-3).
Chiang CL. (1968, ISBN: 978-0-882-75200-6).
Newell C. (1994, ISBN: 978-0-898-62451-9).
Eayres DP, Williams ES (2004) <doi:10.1136/jech.2003.009654>.
Silcocks PBS et al (2001) <doi:10.1136/jech.55.1.38>.
Low and Low (2004) <doi:10.1093/pubmed/fdh175>.
Fingertips Public Health Technical Guide: <https://fingertips.phe.org.uk/profile/guidance/supporting-information/PH-methods/>.

BugReports https://github.com/dhsc-govuk/PHEindicatormethods/issues

Depends R (>= 4.3.0)

License GPL-3

Encoding UTF-8

LazyData true

Imports broom (>= 1.0.0), dplyr (>= 1.1.0), purrr (>= 1.0.0), rlang (>= 1.1.0), stats, tibble (>= 3.2.0), tidyselect (>= 1.2.0), tidyr (>= 1.3.0), lifecycle

Suggests knitr (>= 1.48), readxl (>= 1.4.0), rmarkdown (>= 2.28), testthat (>= 3.0.0), withr (>= 3.0.0)

RoxygenNote 7.3.3

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Georgina Anderson [aut, cre],
Sebastian Fox [ctb],
Matthew Francis [ctb],
Emma Clegg [ctb],
Annabel Westermann [ctb],
Joshua Woolner [ctb],
Charlotte Fellows [ctb],
Olivia Box Power [ctb],
Paul Fryers [rev],
Department of Health and Social Care [cph]

Maintainer Georgina Anderson <georgina.anderson@dhsc.gov.uk>

Repository CRAN

Date/Publication 2025-11-18 15:20:02 UTC

Contents

assign_funnel_significance . . . 3
calculate_dsr . . . 4
calculate_funnel_limits . . . 7
calculate_funnel_points . . . 9
calculate_ISRate . . . 10
calculate_ISRatio . . . 13
DSR_data . . . 15
esp2013 . . . 16
LE_data . . . 17
phe_life_expectancy . . . 17
phe_mean . . . 20
phe_proportion . . . 22
phe_quantile . . . 23
phe_rate . . . 25
phe_sii . . . 26
prevalence_data . . . 30

Index **32**

`assign_funnel_significance` *Identifies whether each value in a dataset falls outside of 95 and/or 99.8 percent control limits based on the aggregated average value across the whole dataset as an indicator of statistically significant difference.*

Description

This follows the funnel plot methodology published on the PHE Fingertips Technical Guidance page: https://fingertips.phe.org.uk/profile/guidance/supporting-information/PH-methods

Usage

```
assign_funnel_significance(
  data,
  numerator,
  denominator,
  rate,
  statistic = NULL,
  rate_type = NULL,
  multiplier = NULL
)
```

Arguments

`data`	a data.frame containing the data to assign significance for; unquoted string; no default
`numerator`	field name from data containing the observed numbers of cases in the sample meeting the required condition (the numerator or observed counts for the control limits); unquoted string; no default
`denominator`	field name from data containing the population(s) in the sample (the denominator or expected counts for the control limits); unquoted string; no default
`rate`	field name from data containing the rate data when creating funnels for a Crude or Directly Standardised Rate; unquoted string; no default
`statistic`	type of statistic to inform funnel calculations. Acceptable values are "proportion", "ratio" or "rate"; string; no default
`rate_type`	if statistic is "rate", specify either "dsr" or "crude"; string; no default
`multiplier`	the multiplier that the rate is normalised with (ie, per 100,000); only required when statistic = "rate"; numeric; no default

Value

returns the original data.frame with the significance level appended

Author(s)

Matthew Francis

See Also

Other PHEindicatormethods package functions: `calculate_ISRate()`, `calculate_ISRatio()`, `calculate_dsr()`, `calculate_funnel_limits()`, `calculate_funnel_points()`, `phe_dsr()`, `phe_life_expectancy()`, `phe_mean()`, `phe_proportion()`, `phe_quantile()`, `phe_rate()`, `phe_sii()`

Examples

```
library(dplyr)
df <- data.frame(
  Area = c("A", "B", "C", "D"),
  numerator = c(10232, 12321, 15123, 13213),
  denominator = c(15232, 16123, 17932, 18475)
)
df %>%
  assign_funnel_significance(numerator, denominator,
                             statistic = "proportion", multiplier = 100)
```

`calculate_dsr` *Calculate Directly Standardised Rates using calculate_dsr*

Description

Calculates directly standardised rates with confidence limits using Byar's method (1) with Dobson method adjustment (2) including option to further adjust confidence limits for non-independent events (3).

Usage

```
calculate_dsr(
  data,
  x,
  n,
  stdpop = NULL,
  type = "full",
  confidence = 0.95,
  multiplier = 1e+05,
  independent_events = TRUE,
  eventfreq = NULL,
  ageband = NULL
)
```

Arguments

`data`	data frame containing the data to be standardised, pre-grouped if multiple DSRs required; unquoted string; no default
`x`	field name from data containing the observed number of events for each standardisation category (eg ageband) within each grouping set (eg area); unquoted string; no default
`n`	field name from data containing the populations for each standardisation category (eg ageband) within each grouping set (eg area); unquoted string; no default
`stdpop`	field name from data containing the standard populations for each age band; unquoted string; no default
`type`	defines the data and metadata columns to be included in output; can be "value", "lower", "upper", "standard" (for all data) or "full" (for all data and metadata); quoted string; default = "full"
`confidence`	the required level of confidence expressed as a number between 0.9 and 1 or a number between 90 and 100 or can be a vector of 0.95 and 0.998, for example, to output both 95 percent and 99.8 percent percent CIs; numeric; default 0.95
`multiplier`	the multiplier used to express the final values (eg 100,000 = rate per 100,000); numeric; default 100,000
`independent_events`	whether events are independent. Set to TRUE for independent events. When set to FALSE an adjustment is made to the confidence intervals - to do this, the dataset provided must include event frequency breakdowns and column x is redefined as the number of unique individuals who experienced each frequency of event, rather than the total number of events.
`eventfreq`	field name from data containing the event frequencies. Only required when independent_events = FALSE; unquoted string; default NULL
`ageband`	field name from data containing the age bands for standardisation. Only required when independent_events = FALSE; unquoted string; default NULL

Value

When type = "full", returns a tibble of total counts, total populations, directly standardised rates, lower confidence limits, upper confidence limits, confidence level, statistic and method for each grouping set. Use the type argument to limit the columns output.

Notes

For total counts >= 10 Byar's method (1) is applied using the internal byars_lower and byars_upper functions. When the total count is < 10 DSRs are not reliable and will therefore be suppressed in the output.

References

(1) Breslow NE, Day NE. Statistical methods in cancer research, volume II: The design and analysis of cohort studies. Lyon: International Agency for Research on Cancer, World Health Organisation; 1987.

(2) Dobson A et al. Confidence intervals for weighted sums of Poisson parameters. Stat Med 1991;10:457-62.

(3) See the DSR chapter of the Fingertips Public Health Technical Guidance

See Also

Other PHEindicatormethods package functions: assign_funnel_significance(), calculate_ISRate(), calculate_ISRatio(), calculate_funnel_limits(), calculate_funnel_points(), phe_dsr(), phe_life_expectancy(), phe_mean(), phe_proportion(), phe_quantile(), phe_rate(), phe_sii()

Examples

```
library(dplyr)
df <- data.frame(
  indicatorid = rep(c(1234, 5678, 91011, 121314), each = 19 * 2 * 5),
  year = rep(2006:2010, each = 19 * 2),
  sex = rep(rep(c("Male", "Female"), each = 19), 5),
  ageband = rep(c(0,5,10,15,20,25,30,35,40,45,
                  50,55,60,65,70,75,80,85,90), times = 10),
  obs = sample(200, 19 * 2 * 5 * 4, replace = TRUE),
  pop = sample(10000:20000, 19 * 2 * 5 * 4, replace = TRUE),
  esp2013 = rep(esp2013, 40)
)

## Example 1 - Default execution
df %>%
  group_by(indicatorid, year, sex) %>%
  calculate_dsr(obs, pop, stdpop = esp2013)

## Example 2 - Calculate both 95% and 99.8% CIs in single execution
df %>%
  group_by(indicatorid, year, sex) %>%
  calculate_dsr(obs, pop, stdpop = esp2013, confidence = c(0.95, 0.998))

## Example 3 - Drop metadata columns from the output
df %>%
  group_by(indicatorid, year, sex) %>%
  calculate_dsr(obs, pop, stdpop = esp2013, type = "standard")

## Example 4 - Calculate DSRs for non-independent events

library(tidyr)

# For non-independent events the input data frame must breakdown events into
# counts of unique individuals by event frequency. The code chunk below
# creates a dummy data frame in this required format. Note that assignment of
# 10%, 20% and 70% of events to each event frequency is purely to create a
# data frame in the required format whilst retaining the same total event and
# population distributions by group and age band as example 1 to allow
# comparison of the outputs.
```

```
df_freq <- df %>%
  mutate(
    f3 = floor((obs * 0.1)/3),         # 10 % of events in individuals with 3 events
    f2 = floor((obs * 0.2)/2),         # 20 % of events in individuals with 2 events
    f1 = (obs - (3 * f3) - (2 * f2))   # 70% of events in individuals with 1 event
  ) %>%
  select(!"obs") %>%
  pivot_longer(
    cols = c("f1", "f2", "f3"),
    names_to = "eventfrequency",
    values_to = "uniqueindividuals",
    names_prefix = "f"
  ) %>%
  mutate(eventfrequency = as.integer(eventfrequency))

# Calculate the dsrs - notice that output DSR values match those in
# example 1 but the confidence intervals are wider

df_freq %>%
  group_by(indicatorid, year, sex) %>%
  calculate_dsr(
    x = uniqueindividuals,
    n = pop,
    stdpop = esp2013,
    independent_events = FALSE,
    eventfreq = eventfrequency,
    ageband = ageband
  )
```

`calculate_funnel_limits` *Calculate control limits for funnel plots*

Description

Calculates control limits adopting a consistent method as per the PHE Fingertips Technical Guidance: https://fingertips.phe.org.uk/profile/guidance/supporting-information/PH-methods

Usage

```
calculate_funnel_limits(
  data,
  numerator,
  denominator,
  rate,
  type = "full",
```

```
  multiplier = NULL,
  statistic = NULL,
  ratio_type = NULL,
  rate_type = NULL,
  years_of_data = NULL
)
```

Arguments

data	a data.frame containing the data to calculate control limits for; unquoted string; no default
numerator	field name from data containing the observed numbers of cases in the sample meeting the required condition (the numerator or observed counts for the control limits); unquoted string; no default
denominator	field name from data containing the population(s) in the sample (the denominator or expected counts for the control limits); unquoted string; no default
rate	field name from data containing the rate data when creating funnels for a Crude or Directly Standardised Rate; unquoted string; no default
type	defines the data and metadata columns to be included in output; "standard" (for all data) or "full" (for all data and metadata); quoted string; default = "full"
multiplier	the multiplier used to express the final values (eg 100 = percentage); numeric; no default
statistic	type of statistic to inform funnel calculations. Acceptable values are "proportion", "ratio" or "rate"; string; no default
ratio_type	if statistic is "ratio", specify either "count" or "isr" (indirectly standardised ratio); string; no default
rate_type	if statistic is "rate", specify either "dsr" or "crude"; string; no default
years_of_data	number of years the data represents; this is required for statistic = "rate"; numeric; no default

Value

returns the original data.frame with the following appended: lower 0.025 limit, upper 0.025 limit, lower 0.001 limit, upper 0.001 limit and baseline average

Author(s)

Matthew Francis

See Also

Other PHEindicatormethods package functions: assign_funnel_significance(), calculate_ISRate(), calculate_ISRatio(), calculate_dsr(), calculate_funnel_points(), phe_dsr(), phe_life_expectancy(), phe_mean(), phe_proportion(), phe_quantile(), phe_rate(), phe_sii()

Examples

```
library(dplyr)
set.seed(123)
df <- data.frame(obs = sample(200, 19 * 2 * 5 * 4, replace = TRUE),
                 pop = sample(10000:20000, 19 * 2 * 5 * 4, replace = TRUE))
df %>%
    calculate_funnel_limits(obs, pop, statistic = "proportion", multiplier = 100)
```

`calculate_funnel_points` *For rate-based funnels: Derive rate and annual population values for charting based. Process removes rates where the rate type is dsr and the number of observed events are below 10.*

Description

For rate-based funnels: Derive rate and annual population values for charting based. Process removes rates where the rate type is dsr and the number of observed events are below 10.

Usage

```
calculate_funnel_points(
  data,
  numerator,
  denominator,
  rate,
  rate_type = NULL,
  years_of_data = NULL,
  multiplier = NULL
)
```

Arguments

`data`	a data.frame containing the data to calculate control limits for; unquoted string; no default
`numerator`	field name from data containing the observed numbers of cases in the sample meeting the required condition (the numerator or observed counts for the control limits); unquoted string; no default
`denominator`	field name from data containing the population(s) in the sample (the denominator or expected counts for the control limits); unquoted string; no default
`rate`	field name from data containing the rate data when creating funnels for a Crude or Directly Standardised Rate; unquoted string; no default
`rate_type`	if statistic is "rate", specify either "dsr" or "crude"; string; no default
`years_of_data`	number of years the data represents; this is required for statistic = "rate"; numeric; no default

multiplier — the multiplier used to express the final values (eg 100 = percentage); numeric; no default

Value

returns the same table as provided with two additional fields. First will have the same name as the rate field, with the suffix "_chart", the second will be called denominator_derived

Author(s)

Sebastian Fox, <sebastian.fox@phe.gov.uk>

See Also

Other PHEindicatormethods package functions: assign_funnel_significance(), calculate_ISRate(), calculate_ISRatio(), calculate_dsr(), calculate_funnel_limits(), phe_dsr(), phe_life_expectancy(), phe_mean(), phe_proportion(), phe_quantile(), phe_rate(), phe_sii()

---

calculate_ISRate *Calculate Indirectly Standardised Rates using calculate_ISRate*

Description

Calculates indirectly standardised rates with confidence limits using Byar's (1) or exact (2) CI method.

Usage

```
calculate_ISRate(
  data,
  x,
  n,
  x_ref,
  n_ref,
  refpoptype = "vector",
  type = "full",
  confidence = 0.95,
  multiplier = 1e+05,
  observed_totals = NULL
)
```

Arguments

data — data.frame containing the data to be standardised, pre-grouped if multiple ISRs required; unquoted string; no default

x	field name from data containing the observed number of events for each standardisation category (eg ageband) within each grouping set (eg area). Alternatively, if not providing age breakdowns for observed events, field name from observed_totals containing the observed number of events within each grouping set ; unquoted string; no default
n	field name from data containing the populations for each standardisation category (eg ageband) within each grouping set (eg area); unquoted string; no default
x_ref	the observed number of events in the reference population for each standardisation category (eg age band); unquoted string referencing a numeric vector or field name from data depending on value of refpoptype; no default
n_ref	the reference population for each standardisation category (eg age band); unquoted string referencing a numeric vector or field name from data depending on value of refpoptype; no default
refpoptype	whether x_ref and n_ref have been specified as vectors or a field name from data; quoted string "field" or "vector"; default = "vector"
type	defines the data and metadata columns to be included in output; can be "value", "lower", "upper", "standard" (for all data) or "full" (for all data and metadata); quoted string; default = "full"
confidence	the required level of confidence expressed as a number between 0.9 and 1 or a number between 90 and 100 or can be a vector of 0.95 and 0.998, for example, to output both 95 percent and 99.8 percent percent CIs; numeric; default 0.95
multiplier	the multiplier used to express the final values (eg 100,000 = rate per 100,000); numeric; default 100,000
observed_totals	data.frame containing total observed events for each group, if not provided with age-breakdowns in data. Must only contain the count field (x) plus grouping columns required to join to data using the same grouping column names; default = NULL

Value

When type = "full", returns a tibble of observed events, expected events, indirectly standardised rate, lower confidence limit, upper confidence limit, confidence level, statistic and method for each grouping set

Notes

User MUST ensure that x, n, x_ref and n_ref vectors are all ordered by the same standardisation category values as records will be matched by position.

For numerators >= 10 Byar's method (1) is applied using the internal byars_lower and byars_upper functions. For small numerators Byar's method is less accurate and so an exact method (2) based on the Poisson distribution is used.

This function directly replaced phe_isr which was fully deprecated in package version 2.0.0 due to ambiguous naming

References

(1) Breslow NE, Day NE. Statistical methods in cancer research, volume II: The design and analysis of cohort studies. Lyon: International Agency for Research on Cancer, World Health Organisation; 1987.

(2) Armitage P, Berry G. Statistical methods in medical research (4th edn). Oxford: Blackwell; 2002.

See Also

Other PHEindicatormethods package functions: assign_funnel_significance(), calculate_ISRatio(), calculate_dsr(), calculate_funnel_limits(), calculate_funnel_points(), phe_dsr(), phe_life_expectancy(), phe_mean(), phe_proportion(), phe_quantile(), phe_rate(), phe_sii()

Examples

```
library(dplyr)
df <- data.frame(indicatorid = rep(c(1234, 5678, 91011, 121314), each = 19 * 2 * 5),
                 year = rep(2006:2010, each = 19 * 2),
                 sex = rep(rep(c("Male", "Female"), each = 19), 5),
                 ageband = rep(c(0,5,10,15,20,25,30,35,40,45,
                                 50,55,60,65,70,75,80,85,90), times = 10),
                 obs = sample(200, 19 * 2 * 5 * 4, replace = TRUE),
                 pop = sample(10000:20000, 19 * 2 * 5 * 4, replace = TRUE))

refdf <- data.frame(refcount = sample(200, 19, replace = TRUE),
                    refpop = sample(10000:20000, 19, replace = TRUE))

## calculate multiple ISRs in single execution
df %>%
    group_by(indicatorid, year, sex) %>%
    calculate_ISRate(obs, pop, refdf$refcount, refdf$refpop)

## execute without outputting metadata fields
df %>%
    group_by(indicatorid, year, sex) %>%
   calculate_ISRate(obs, pop, refdf$refcount, refdf$refpop, type="standard", confidence=99.8)

## calculate 95% and 99.8% CIs in single execution
df %>%
    group_by(indicatorid, year, sex) %>%
    calculate_ISRate(obs, pop, refdf$refcount, refdf$refpop, confidence = c(0.95, 0.998))

## Calculate ISR when observed totals aren't available with age-breakdowns
observed_totals <- data.frame(indicatorid = rep(c(1234, 5678, 91011, 121314), each = 10),
                      year = rep(rep(2006:2010, each = 2),4),
                      sex = rep(rep(c("Male", "Female"), each = 1),20),
                      observed = sample(1500:2500, 40))

df %>%
    group_by(indicatorid, year, sex) %>%
```

```
calculate_ISRate(observed, pop, refdf$refcount, refdf$refpop,
observed_totals = observed_totals)
```

`calculate_ISRatio` *Calculate Indirectly standardised ratios using calculate_ISRatio*

Description

Calculates standard mortality ratios (or indirectly standardised ratios) with confidence limits using Byar's (1) or exact (2) CI method.

Usage

```
calculate_ISRatio(
  data,
  x,
  n,
  x_ref,
  n_ref,
  refpoptype = "vector",
  type = "full",
  confidence = 0.95,
  refvalue = 1,
  observed_totals = NULL
)
```

Arguments

`data`	data.frame containing the data to be standardised, pre-grouped if multiple ISRs required; unquoted string; no default
`x`	field name from data containing the observed number of events for each standardisation category (eg ageband) within each grouping set (eg area). Alternatively, if not providing age breakdowns for observed events, field name from observed_totals containing the observed number of events within each grouping set ; unquoted string; no default
`n`	field name from data containing the populations for each standardisation category (eg ageband) within each grouping set (eg area); unquoted string; no default
`x_ref`	the observed number of events in the reference population for each standardisation category (eg age band); unquoted numeric vector or field name from data depending on value of refpoptype; no default
`n_ref`	the reference population for each standardisation category (eg age band); unquoted numeric vector or field name from data depending on value of refpoptype; no default

`refpoptype`	whether x_ref and n_ref have been specified as vectors or a field name from data; quoted string "field" or "vector"; default = "vector"
`type`	defines the data and metadata columns to be included in output; can be "value", "lower", "upper", "standard" (for all data) or "full" (for all data and metadata); quoted string; default = "full"
`confidence`	the required level of confidence expressed as a number between 0.9 and 1 or a number between 90 and 100 or can be a vector of 0.95 and 0.998, for example, to output both 95 percent and 99.8 percent percent CIs; numeric; default 0.95
`refvalue`	the standardised reference ratio, numeric, default = 1
`observed_totals`	data.frame containing total observed events for each group, if not provided with age-breakdowns in data. Must only contain the count field (x) plus grouping columns required to join to data using the same grouping column names; default = NULL

Value

When type = "full", returns a tibble of observed events, expected events, standardised mortality ratios, lower confidence limits, upper confidence limits, confidence level, statistic and method for each grouping set

Notes

User MUST ensure that x, n, x_ref and n_ref vectors are all ordered by the same standardisation category values as records will be matched by position.

For numerators >= 10 Byar's method (1) is applied using the internal byars_lower and byars_upper functions. For small numerators Byar's method is less accurate and so an exact method (2) based on the Poisson distribution is used.

This function directly replaced phe_smr which was fully deprecated in package version 2.0.0 due to ambiguous naming

References

(1) Breslow NE, Day NE. Statistical methods in cancer research, volume II: The design and analysis of cohort studies. Lyon: International Agency for Research on Cancer, World Health Organisation; 1987.

(2) Armitage P, Berry G. Statistical methods in medical research (4th edn). Oxford: Blackwell; 2002.

See Also

Other PHEindicatormethods package functions: `assign_funnel_significance()`, `calculate_ISRate()`, `calculate_dsr()`, `calculate_funnel_limits()`, `calculate_funnel_points()`, `phe_dsr()`, `phe_life_expectancy()`, `phe_mean()`, `phe_proportion()`, `phe_quantile()`, `phe_rate()`, `phe_sii()`

Examples

```
library(dplyr)
df <- data.frame(indicatorid = rep(c(1234, 5678, 91011, 121314), each = 19 * 2 * 5),
                 year = rep(2006:2010, each = 19 * 2),
                 sex = rep(rep(c("Male", "Female"), each = 19), 5),
                 ageband = rep(c(0,5,10,15,20,25,30,35,40,45,
                                 50,55,60,65,70,75,80,85,90), times = 10),
                 obs = sample(200, 19 * 2 * 5 * 4, replace = TRUE),
                 pop = sample(10000:20000, 19 * 2 * 5 * 4, replace = TRUE))

refdf <- data.frame(refcount = sample(200, 19, replace = TRUE),
                    refpop = sample(10000:20000, 19, replace = TRUE))

df %>%
    group_by(indicatorid, year, sex) %>%
    calculate_ISRatio(obs, pop, refdf$refcount, refdf$refpop, type="standard")

## OR

df %>%
    group_by(indicatorid, year, sex) %>%
   calculate_ISRatio(obs, pop, refdf$refcount, refdf$refpop, confidence=99.8, refvalue=100)

## Calculate ISR when observed totals aren't available with age-breakdowns
observed_totals <- data.frame(indicatorid = rep(c(1234, 5678, 91011, 121314), each = 10),
                      year = rep(rep(2006:2010, each = 2),4),
                      sex = rep(rep(c("Male", "Female"), each = 1),20),
                      observed = sample(1500:2500, 40))

df %>%
    group_by(indicatorid, year, sex) %>%
    calculate_ISRatio(observed, pop, refdf$refcount, refdf$refpop,
    observed_totals = observed_totals)
```

DSR_data *SII test datasets - DSR*

Description

A data table of dummy Directly Standardised Rates by deprivation quintiles

Usage

```
data(DSR_data)
```

Format

A data table

Examples

```
DSR_data
```

`esp2013` *European Standard Population 2013*

Description

A numeric vector containing nineteen 5-year age band populations making up the 2013 European Standard Population ordered from age 0-4, 5-9, 10-14 ... to ... 85-89, 90+. Sorted by increasing age band.

Usage

```
esp2013
```

Format

A numeric vector with 19 elements

Value

5000 5500 5500 5500 6000 6000 6500 7000 7000 7000 7000 6500 6000 5500 5000 4000 2500 1500 1000

Notes

The 2013 European Standard Population is modelled and published by Eurostat (1) for use in the production of age-standardised rates. It uses the unweighted average 2010-based population projections of the European Union (x27) and European Free Trade Association (x4) countries for the period 2011-2030 broken down into 5-year age bands from age 0 - age 95+ with the 0-5 age band separated into age 0 and age 1-4. The version provided with this package combines the age 0 and age 1-4 populations into a single 0-4 age band and combines the 90-94 and 95+ populations into a single 90+ age band, giving 19 age bands in total.

References

(1) Eurostat Methodologies and Working Papers. Revision of the European Standard Population: Report of Eurostat's Taskforce, 2013.
https://ec.europa.eu/eurostat/documents/3859598/5926869/KS-RA-13-028-EN.PDF/e713fa79-1add-44e8-b23c

Examples

```
esp2013
```

LE_data *SII test datasets - Life Expectancy*

Description

A data table of life expectancy data by area and deprivation decile

Usage

```
data(LE_data)
```

Format

A data table

Examples

```
LE_data
```

phe_life_expectancy *Calculate Life Expectancy using phe_life_expectancy*

Description

Compute life expectancy for a given age, and its standard error

Usage

```
phe_life_expectancy(
  data,
  deaths,
  population,
  startage,
 age_contents = c(0L, 1L, 5L, 10L, 15L, 20L, 25L, 30L, 35L, 40L, 45L, 50L, 55L, 60L,
    65L, 70L, 75L, 80L, 85L, 90L),
  le_age = "all",
  type = "full",
  confidence = 0.95
)
```

Arguments

data	data.frame or tbl containing the deaths and population data
deaths	field name from data containing the number of deaths within age band; unquoted string; no default
population	field name from data containing the population within age band; unquoted string; no default
startage	field name from data containing the age band; no default
age_contents	vector; describes the contents of startage in the ascending order. This vector is used to check whether each group in data contains the complete set of age bands for the calculation to occur. It is also used to reorder the data based on the startage field
le_age	the age band to return the life expectancy for. The default is "all", where the function returns the life expectancy values for all ages appended onto the input table. Any other value (or vector of values) must be age bands described by the age_contents input
type	type of output; can be "standard" or "full" (full contains added details on the calculation within the dataframe); quoted string; default full
confidence	the required level of confidence expressed as a number between 0.9 and 1 or a number between 90 and 100 or can be a vector of 0.95 and 0.998, for example, to output both 95 percent and 99.8 percent percent CIs; numeric; default 0.95

Details

This function aligns with the methodology in Public Health England's Life Expectancy Calculator available on the Fingertips Technical Guidance web page.

The function is for an abridged life table using 5 year age intervals with a final age interval of 90+. The table has been completed using the methods described by Chiang.(1, 2) This age structure and methodology is used by The Office for National Statistics to produce life expectancy at national and local authority level.(3)

This function includes an adjustment to the method for calculating the variance of the life expectancy estimate to include a term for the variance associated with the final age interval. In the Chiang method the variance of the life expectancy is the weighted sum of the variance of the probability of survival across all the age intervals. For the final age interval the probability of survival is, Chiang argues, zero and has zero variance. However, Silcocks et al argue(4) that in the case of the final age interval the life expectancy is dependent not on the probability of survival but on the mean length of survival $(1/M < sub > omega < /sub >)$. Therefore the variance associated with the final age interval depends on the age-specific mortality rate $M < sub > omega < /sub >$.

Life expectancy cannot be calculated if the person-years in any given age interval is zero. It will also not be calculated if the total person-years is less than 5,000 as this is considered to be the minimum size for robust calculation of life expectancy.(5) Zero death counts are not a problem, except for the final age interval - there must be at least one death in the 90+ interval for the calculations to be possible.

Individual Life Expectancy values will be suppressed (although confidence intervals will be shown) when the 95% confidence interval is greater than 20 years.

The methodology used in this function, along with discussion of alternative options for life expectancy calculation for small areas, were described Eayres and Williams.(6)

Value

returns a data frame containing the life expectancies and confidence intervals for each le_age requested. When type = 'full' additionally returns the cumulative populations and deaths used in each LE calculation and metadata indicating parameters passed.

Author(s)

Sebastian Fox, <sebastian.fox@phe.gov.uk>

References

(1) Chiang CL. The Life Table and its Construction. In: Introduction to Stochastic Processes in Biostatistics. New York, John Wiley & Sons, 1968:189-214.

(2) Newell C. Methods and Models in Demography. Chichester, John Wiley & Sons, 1994:63-81

(3) Office for National Statistics Report. Life expectancy at birth by health and local authorities in the United Kingdom, 1998 to 2000 (3-year aggregate figures.) Health Statistics Quarterly 2002;13:83-90

(4) Silcocks PBS, Jenner DA, Reza R. Life expectancy as a summary of mortality in a population: statistical considerations and suitability for use by health authorities. J Epidemiol Community Health 2001;55:38-43

(5) Toson B, Baker A. Life expectancy at birth: methodological options for small populations. National Statistics Methodological Series No 33. HMSO 2003.

(6) Eayres DP, Williams ES. Evaluation of methodologies for small area life expectancy estimation. J Epidemiol Community Health 2004;58:243-249

See Also

Other PHEindicatormethods package functions: assign_funnel_significance(), calculate_ISRate(), calculate_ISRatio(), calculate_dsr(), calculate_funnel_limits(), calculate_funnel_points(), phe_dsr(), phe_mean(), phe_proportion(), phe_quantile(), phe_rate(), phe_sii()

Examples

```
library(dplyr)

## A simple example
df <- data.frame(startage = c(0L, 1L, 5L, 10L, 15L, 20L, 25L, 30L, 35L, 40L, 45L, 50L, 55L,
                              60L, 65L, 70L, 75L, 80L, 85L, 90L),
                 pops = c(7060L, 35059L, 46974L, 48489L, 43219L, 38561L, 46009L, 57208L,
                          61435L, 55601L, 50209L, 56416L, 46411L, 39820L, 37978L,
                          37039L, 33288L, 23306L, 11936L, 11936L),
                 deaths = c(17L, 9L, 4L, 8L, 20L, 15L, 24L, 33L, 50L, 71L, 100L, 163L,
```

```
                            263L, 304L, 536L, 872L, 1390L, 1605L, 1936L, 1937L))
phe_life_expectancy(df, deaths, pops, startage)

## or with multiple confidence limits
phe_life_expectancy(df, deaths, pops, startage, confidence = c(95, 99.8))

## OR

phe_life_expectancy(df, deaths, pops, startage, le_age = c(5, 25), type = "standard")

## Unordered age bands example
df <- data.frame(startage = c("0", "1-4", "5-9", "10 - 14", "15 - 19", "20 - 24", "25 - 29",
                              "30 - 34", "35 - 39", "40 - 44", "45 - 49", "50 - 54",
                              "55 - 59", "60 - 64", "65 - 69", "75 - 79", "80 - 84",
                              "85 - 89", "90 +", "70 - 74"),
                 pops = c(7060L, 35059L, 46974L, 48489L, 43219L, 38561L, 46009L, 57208L,
                          61435L, 55601L, 50209L, 56416L, 46411L, 39820L, 37039L,
                          23306L, 11936L, 11936L, 37978L, 33288L),
                 deaths = c(17L, 9L, 4L, 8L, 20L, 15L, 24L, 33L, 50L, 71L, 100L, 163L,
                            263L, 304L, 872L, 1605L, 1936L, 1937L, 536L, 1390L))
phe_life_expectancy(df, deaths, pops, startage,
                    age_contents = c("0", "1-4", "5-9",
                                     "10 - 14", "15 - 19",
                                     "20 - 24", "25 - 29",
                                     "30 - 34", "35 - 39",
                                     "40 - 44", "45 - 49",
                                     "50 - 54", "55 - 59",
                                     "60 - 64", "65 - 69",
                                     "70 - 74", "75 - 79",
                                     "80 - 84", "85 - 89",
                                     "90 +"))

df <- data.frame(area = c(rep("Area 1", 20), rep("Area 2", 20)),
              startage = rep(c(0L, 1L, 5L, 10L, 15L, 20L, 25L, 30L, 35L, 40L, 45L, 50L, 55L,
                               60L, 65L, 70L, 75L, 80L, 85L, 90L), 2),
               pops = rep(c(7060L, 35059L, 46974L, 48489L, 43219L, 38561L, 46009L, 57208L,
                            61435L, 55601L, 50209L, 56416L, 46411L, 39820L, 37978L,
                            37039L, 33288L, 23306L, 11936L, 11936L), 2),
                 deaths = rep(c(17L, 9L, 4L, 8L, 20L, 15L, 24L, 33L, 50L, 71L, 100L, 163L,
                                263L, 304L, 536L, 872L, 1390L, 1605L, 1936L, 1937L), 2))
df %>%
       group_by(area) %>%
       phe_life_expectancy(deaths, pops, startage)
```

---

phe_mean *Calculate Means using phe_mean*

Description

Calculates means with confidence limits using Student's t-distribution method.

Usage

```
phe_mean(data, x, type = "full", confidence = 0.95)
```

Arguments

data	a data.frame containing the data to calculate means for, pre-grouped if multiple means required; unquoted string; no default
x	field name from data containing the values to calculate the means for; unquoted string; no default
type	defines the data and metadata columns to be included in output; can be "value", "lower", "upper", "standard" (for all data) or "full" (for all data and metadata); quoted string; default = "full"
confidence	the required level of confidence expressed as a number between 0.9 and 1 or a number between 90 and 100 or can be a vector of 0.95 and 0.998, for example, to output both 95 percent and 99.8 percent percent CIs; numeric; default 0.95

Value

When type = "full", returns a data.frame of value_sum, value_count, stdev, value, lowercl, uppercl, confidence, statistic and method for each grouping set

See Also

Other PHEindicatormethods package functions: assign_funnel_significance(), calculate_ISRate(), calculate_ISRatio(), calculate_dsr(), calculate_funnel_limits(), calculate_funnel_points(), phe_dsr(), phe_life_expectancy(), phe_proportion(), phe_quantile(), phe_rate(), phe_sii()

Examples

```
library(dplyr)
df <- data.frame(values = c(30,40,50,60))

## default execution
phe_mean(df, values)

## calculate 95% and 99.8% CIs in single execution
phe_mean(df, values, confidence = c(0.95, 0.998))

## calculate multiple means in a single execution

df2 <- data.frame(area = rep(c("Area1", "Area2"),each=3),
                  values = c(20,30,40,200,300,400)) %>%
                  group_by(area)
phe_mean(df2,values)
phe_mean(df2,values,type="standard", confidence=0.998)
```

phe_proportion *Calculate Proportions using phe_proportion*

Description

Calculates proportions with confidence limits using Wilson Score method (1,2).

Usage

```
phe_proportion(data, x, n, type = "full", confidence = 0.95, multiplier = 1)
```

Arguments

data	a data.frame containing the data to calculate proportions for, pre-grouped if proportions required for group aggregates; unquoted string; no default
x	field name from data containing the observed numbers of cases in the sample meeting the required condition (the numerator for the proportion); unquoted string; no default
n	field name from data containing the number of cases in the sample (the denominator for the proportion); unquoted string; no default
type	defines the data and metadata columns to be included in output; can be "value", "lower", "upper", "standard" (for all data) or "full" (for all data and metadata); quoted string; default = "full"
confidence	the required level of confidence expressed as a number between 0.9 and 1 or a number between 90 and 100. The vector c(0.95, 0.998) can also be passed to output both 95 percent and 99.8 percent CIs, or an NA value can be passed if no confidence intervals are required.; numeric; default 0.95
multiplier	the multiplier used to express the final values (eg 100 = percentage); numeric; default 1

Value

When type = "full", returns the original data.frame with the following appended: proportion, lower confidence limit, upper confidence limit, confidence level, statistic and method

Notes

Wilson Score method (2) is applied using the internal wilson_lower and wilson_upper functions.

The percentage argument was deprecated in v1_1_0, please use multiplier argument instead

References

(1) Wilson EB. Probable inference, the law of succession, and statistical inference. J Am Stat Assoc; 1927; 22. Pg 209 to 212.
(2) Newcombe RG, Altman DG. Proportions and their differences. In Altman DG et al. (eds). Statistics with confidence (2nd edn). London: BMJ Books; 2000. Pg 46 to 48.

See Also

Other PHEindicatormethods package functions: assign_funnel_significance(), calculate_ISRate(), calculate_ISRatio(), calculate_dsr(), calculate_funnel_limits(), calculate_funnel_points(), phe_dsr(), phe_life_expectancy(), phe_mean(), phe_quantile(), phe_rate(), phe_sii()

Examples

```
# ungrouped data frame
df <- data.frame(area = rep(c("Area1","Area2","Area3","Area4"), each=3),
                 numerator = c(NA,82,9,48, 6500,8200,10000,10000,8,7,750,900),
                 denominator = rep(c(100,10000,10000,10000), each=3))

phe_proportion(df, numerator, denominator)
phe_proportion(df, numerator, denominator, confidence=99.8)
phe_proportion(df, numerator, denominator, type="standard")
phe_proportion(df, numerator, denominator, confidence = c(0.95, 0.998))

# grouped data frame
library(dplyr)
dfg <- df |> group_by(area)
phe_proportion(dfg, numerator, denominator, multiplier=100)
```

---

phe_quantile *Assign Quantiles using phe_quantile*

Description

Assigns data to quantiles based on numeric data rankings.

Usage

```
phe_quantile(
  data,
  values,
  nquantiles = 10L,
  invert = TRUE,
  inverttype = "logical",
  type = "full"
)
```

Arguments

data	a data frame containing the quantitative data to be assigned to quantiles. If pre-grouped, separate sets of quantiles will be assigned for each grouping set; unquoted string; no default

values	field name from data containing the numeric values to rank data by and assign quantiles from; unquoted string; no default
nquantiles	the number of quantiles to separate each grouping set into; numeric; default=10L
invert	whether the quantiles should be directly (FALSE) or inversely (TRUE) related to the numerical value order; logical (to apply same value to all grouping sets) OR unquoted string referencing field name from data that stores logical values for each grouping set; default = TRUE (ie highest values assigned to quantile 1)
inverttype	whether the invert argument has been specified as a logical value or a field name from data; quoted string "field" or "logical"; default = "logical"
type	defines whether to include metadata columns in output to reference the arguments passed; can be "standard" or "full"; quoted string; default = "full"

Value

When type = "full", returns the original data.frame with quantile (quantile value), nquantiles (number of quantiles requested), groupvars (grouping sets quantiles assigned within) and invert (indicating direction of quantile assignment) fields appended.

Notes

See OHID Technical Guide - Assigning Deprivation Categories for methodology. In particular, note that this function strictly applies the algorithm defined but some manual review, and potentially adjustment, is advised in some cases where multiple small areas with equal rank fall across a natural quantile boundary.

See Also

Other PHEindicatormethods package functions: assign_funnel_significance(), calculate_ISRate(), calculate_ISRatio(), calculate_dsr(), calculate_funnel_limits(), calculate_funnel_points(), phe_dsr(), phe_life_expectancy(), phe_mean(), phe_proportion(), phe_rate(), phe_sii()

Examples

```
df <- data.frame(
  region    = as.character(rep(c("Region1","Region2","Region3","Region4"),
                               each=250)),
  smallarea = as.character(paste0("Area",seq_along(1:1000))),
  vals      = as.numeric(sample(200, 1000, replace = TRUE)),
  stringsAsFactors = FALSE)

# assign small areas to deciles across whole data frame
phe_quantile(df, vals)

# assign small areas to deciles within regions by pre-grouping the data frame
library(dplyr)
df_grp <- df %>% group_by(region)
phe_quantile(df_grp, vals)

# assign small areas to quintiles, where highest value = highest quantile
phe_quantile(df, vals, nquantiles = 5L, invert=FALSE)
```

phe_rate *Calculate Rates using phe_rate*

Description

Calculates rates with confidence limits using Byar's (1) or exact (2) CI method.

Usage

```
phe_rate(data, x, n, type = "full", confidence = 0.95, multiplier = 1e+05)
```

Arguments

data	the data.frame containing the data to calculate rates for, pre-grouped if proportions required for group aggregates; unquoted string; no default
x	field name from data containing the rate numerators (eg observed number of events); unquoted string; no default
n	field name from data containing the rate denominators (eg populations); unquoted string; no default
type	defines the data and metadata columns to be included in output; can be "value", "lower", "upper", "standard" (for all data) or "full" (for all data and metadata); quoted string; default = "full"
confidence	the required level of confidence expressed as a number between 0.9 and 1 or a number between 90 and 100 or can be a vector of 0.95 and 0.998, for example, to output both 95 percent and 99.8 percent percent CIs; numeric; default 0.95
multiplier	the multiplier used to express the final values (eg 100,000 = rate per 100,000); numeric; default 100,000

Value

When type = "full", returns the original data.frame with the following appended: rate, lower confidence limit, upper confidence limit, confidence level, statistic and method

Notes

For numerators >= 10 Byar's method (1) is applied using the internal byars_lower and byars_upper functions. For small numerators Byar's method is less accurate and so an exact method (2) based on the Poisson distribution is used.

References

(1) Breslow NE, Day NE. Statistical methods in cancer research, volume II: The design and analysis of cohort studies. Lyon: International Agency for Research on Cancer, World Health Organisation; 1987.
(2) Armitage P, Berry G. Statistical methods in medical research (4th edn). Oxford: Blackwell; 2002.

See Also

Other PHEindicatormethods package functions: assign_funnel_significance(), calculate_ISRate(), calculate_ISRatio(), calculate_dsr(), calculate_funnel_limits(), calculate_funnel_points(), phe_dsr(), phe_life_expectancy(), phe_mean(), phe_proportion(), phe_quantile(), phe_sii()

Examples

```
# ungrouped data frame
df <- data.frame(area = rep(c("Area1","Area2","Area3","Area4"), each=3),
                 obs = c(NA,82,9,48, 6500,8200,10000,10000,8,7,750,900),
                 pop = rep(c(100,10000,10000,10000), each=3))

phe_rate(df, obs, pop)
phe_rate(df, obs, pop, type="standard")
phe_rate(df, obs, pop, confidence=99.8, multiplier=100)

# grouped data frame
library(dplyr)
dfg <- df %>% group_by(area)
phe_rate(dfg, obs, pop)
```

---

phe_sii *Calculate Slope Index of Inequality using phe_sii*

Description

phe_sii()returns the slope index of inequality (SII) statistic for each subgroup of the inputted dataframe, with lower and upper confidence limits based on the specified confidence.

Usage

```
phe_sii(
  data,
  quantile,
  population,
  x = NULL,
  value = NULL,
  value_type = 0,
  transform = FALSE,
  lower_cl = NULL,
  upper_cl = NULL,
  se = NULL,
  multiplier = 1,
  repetitions = 1e+05,
  confidence = 0.95,
  rii = FALSE,
  intercept = FALSE,
```

```
  reliability_stat = FALSE,
  type = "full"
)
```

Arguments

data	data.frame containing the required input fields, pre-grouped if an SII is required for each subgroup; unquoted string; no default
quantile	field name within data that contains the quantile label (e.g. decile). The number of quantiles should be between 5 and 100; unquoted string; no default
population	field name within data that contains the quantile populations (ie, denominator). Non-zero populations are required for all quantiles to calculate SII for an area; unquoted string; no default
x	(for indicators that are proportions) field name within data that contains the members of the population with the attribute of interest (ie, numerator). This will be divided by population to calculate a proportion as the indicator value (if value field is not provided); unquoted string; no default
value	field name within data that contains the indicator value (this does not need to be supplied for proportions if count and population are given); unquoted string; no default
value_type	indicates the indicator type (1 = rate, 2 = proportion, 0 = other). The value_type argument is used to determine whether data should be transformed prior to calculation of the standard error and/or SII. See the Tansformations section for full details; integer; default 0
transform	option to transform input rates or proportions prior to calculation of the SII. See the Transformations section for full details; logical; default FALSE
lower_cl	field name within data that contains 95 percent lower confidence limit of indicator value (to calculate standard error of indicator value). This field is needed if the se field is not supplied; unquoted string; no default
upper_cl	field name within data that contains 95 percent upper confidence limit of indicator value (to calculate standard error of indicator value). This field is needed if the se field is not supplied; unquoted string; no default
se	field name within data that contains the standard error of the indicator value. If not supplied, this will be calculated from the 95 percent lower and upper confidence limits (i.e. one or the other of these fields must be supplied); unquoted string; no default
multiplier	factor to multiply the SII and SII confidence limits by (e.g. set to 100 to return prevalences on a percentage scale between 0 and 100). If the multiplier is negative, the inverse of the RII is taken to account for the change in polarity; numeric; default 1;
repetitions	number of random samples to perform to return confidence interval of SII (and RII). Minimum is 1000, no maximum (though the more repetitions, the longer the run time); numeric; default 100,000
confidence	confidence level used to calculate the lower and upper confidence limits of SII, expressed as a number between 0.9 and 1, or 90 and 100. It can be a vector of

0.95 and 0.998, for example, to output both 95 percent and 99.8 percent CIs; numeric; default 0.95

`rii` option to return the Relative Index of Inequality (RII) with associated confidence limits as well as the SII; logical; default FALSE

`intercept` option to return the intercept value of the regression line (y value where x=0); logical; default FALSE

`reliability_stat`
option to carry out the SII confidence interval simulation 10 times instead of once and return the Mean Average Difference between the first and subsequent samples (as a measure of the amount of variation). Warning: this will significantly increase run time of the function and should first be tested on a small number of repetitions; logical; default FALSE

`type` "full" output includes columns in the output dataset specifying the parameters the user has input to the function (value_type, multiplier, CI_confidence, CI_method); character string either "full" or "standard"; default "full"

Details

The Relative Index of Inequality (RII) can also be returned via an optional argument.

The SII and RII are two measures of health inequality. They show the relation between the level of health or frequency of a health problem in different population groups and the ranking of these groups on the social scale.

The input dataframe should be grouped before passing to the function if an SII/RII for each subgroup is required, and quantiles ordered from least to most advantaged.

Value

The SII with lower and upper confidence limits for each subgroup of the inputted data.frame.

Calculation

The SII is calculated using linear regression (1). To allow for differences in population size between quantiles (e.g. deprivation deciles), each is given a rank score (or relative rank) based on the midpoint of its range in the cumulative distribution of the total area population. The quantiles are first ordered (e.g from 1 most deprived to 10 least deprived for deprivation deciles). If quantile 1 then contains 12 percent of the total population, its relative rank is `0.12/2=0.6`. If quantile 2 includes 10 percent of the population, its relative rank is `0.12+(0.10/2)=0.17`. A square root transformation is applied to the regression to account for heteroskedasticity (the tendancy for the variances of the quantile values to be related to the size of the values, ie larger values will tend to have larger variances). A regression model is fitted to the transformed data: $Y * \sqrt{a} = \sqrt{a} + b * \sqrt{a}$, where Y is the value of the indicator for the quantile, a is the proportion of the total population in the quantile and b is the relative rank. The SII is the gradient of the resulting fitted line, and could be positive or negative according to the indicator polarity. Since the relative ranks, by definition, range from 0 to 1, the SII is the difference between the fitted value at `x=1` and `x=0`. The RII is the ratio of the fitted value at `x=1,Y1` and the fitted value at `x=0,Y0`. which can be calculated as: `RII = (Y0 + SII)/Y0`

Transformations

The indicator type can be specified as 1 (rate), 2 (proportion) or 0 (other), using the `value_type` parameter. This setting determines the data transformations that will be applied in the following two parts of the method.

Use in conjunction with the `transform` parameter in calculation of the SII: It is recommended that rates and proportions are transformed prior to calculation of the SII by setting the `transform` parameter to TRUE for these indicator types. This will perform a log transformation for rates, or logit for proportions, and return outputs transformed back to the original units of the indicator. These transformations are recommended to improve the linearity between the indicator values and the quantile, which is an assumption of the method. A user-provided standard error will not be accepted when the `transform` parameter is set to TRUE as the confidence limits are required for this transformation.

Use in calculation of the standard error: Rates and proportions, and their confidence limits, are transformed prior to calculation of the standard error for each quantile. This is because it is assumed that the confidence interval around the indicator value is non-symmetric for these indicator types. Note that this transformation is not controlled by the `transform` parameter and is applied based on the value of the `value_type` parameter only. A user-provided standard error will not be accepted when the `transform` parameter is set to TRUE as the confidence limits are required for this transformation.

Warning

The SII calculation assumes a linear relationship between indicator value and quantile. Where this is not the case the transform option should be considered. Small populations within quantiles can make the SII unstable. This function does not include checks for linearity or stability; it is the user's responsibility to ensure the input data is suitable for the SII calculation.

References

(1) Low A & Low A. Measuring the gap: quantifying and comparing local health inequalities. Journal of Public Health; 2004;26:388-395.

See Also

Other PHEindicatormethods package functions: `assign_funnel_significance()`, `calculate_ISRate()`, `calculate_ISRatio()`, `calculate_dsr()`, `calculate_funnel_limits()`, `calculate_funnel_points()`, `phe_dsr()`, `phe_life_expectancy()`, `phe_mean()`, `phe_proportion()`, `phe_quantile()`, `phe_rate()`

Examples

```
library(dplyr)

data <- data.frame(area = c(rep("Area1", 10), rep("Area2", 10)),
                   decile = c(1:10, 1:10),
             population = c(7291, 7997, 6105, 7666, 5790, 6934, 5918, 5974, 7147, 7534, 21675,
                            20065, 19750, 24713, 20112, 19618, 22408, 19752, 18939, 19312),
                value = c(75.9, 78.3, 83.8, 83.6, 80.5, 81.1, 81.7, 84.2, 80.6, 86.3, 70.5,
```

```
                          71.6, 72.5, 73.5, 73.1, 76.2, 78.7, 80.6, 80.9, 80),
                    lowerCL = c(72.7,75.3,80.9,80.2,77.1,78,79,81.4,75.8,83.2,
                                70.1,71.1,72,73.1, 72.7, 75.7, 78.2,80.1,80.4,79.5),
                    upperCL = c(79.1,81.4,86.8,87.1,83.8,84.2,84.4,86.9,85.4,
                                89.4,71,72.1,73.2,73.7,75.8,78.8,79.8,81.2,81.3,80.9),
                    StandardError = c(1.64,1.58,1.51,1.78,1.7,1.56,1.37,1.4,2.43,
                                      1.57,0.23,0.26,0.3,0.16,0.79,0.78,0.4,0.28,0.23,0.35)
                    )


# Run SII function on the two areas in the data
phe_sii(group_by(data, area),
        decile,
        population,
        value_type = 0, # default normal distribution
        value = value,
        lower_cl = lowerCL,
        upper_cl = upperCL,
        confidence = 0.95,
        rii = TRUE,
        type = "standard")

# Supplying the standard error instead of the indicator 95 percent confidence limits
# gives the same result
phe_sii(group_by(data, area),
        decile,
        population,
        value_type = 0,
        value = value,
        se = StandardError,
        confidence = 0.95,
        rii = TRUE,
        type = "standard")

# multiple confidence intervals, log transforming the data if they are rates
phe_sii(group_by(data, area),
        decile,
        population,
        value_type = 1,
        transform = TRUE,
        value = value,
        lower_cl = lowerCL,
        upper_cl = upperCL,
        confidence = c(0.95, 0.998),
        repetitions = 10000,
        rii = TRUE,
        type = "standard")
```

`prevalence_data` *SII test datasets - Prevalence*

Description

A data table of example prevalence data by area and deprivation decile

Usage

```
data(prevalence_data)
```

Format

A data table

Examples

```
prevalence_data
```

Index

∗ **PHEindicatormethods package functions**
  assign_funnel_significance, 3
  calculate_dsr, 4
  calculate_funnel_limits, 7
  calculate_funnel_points, 9
  calculate_ISRate, 10
  calculate_ISRatio, 13
  phe_life_expectancy, 17
  phe_mean, 20
  phe_proportion, 22
  phe_quantile, 23
  phe_rate, 25
  phe_sii, 26
∗ **datasets**
  DSR_data, 15
  esp2013, 16
  LE_data, 17
  prevalence_data, 30

assign_funnel_significance, 3, *6*, *8*, *10*, *12*, *14*, *19*, *21*, *23, 24*, *26*, *29*

calculate_dsr, *4*, 4, *8*, *10*, *12*, *14*, *19*, *21*, *23, 24*, *26*, *29*
calculate_funnel_limits, *4*, *6*, 7, *10*, *12*, *14*, *19*, *21*, *23, 24*, *26*, *29*
calculate_funnel_points, *4*, *6*, *8*, 9, *12*, *14*, *19*, *21*, *23, 24*, *26*, *29*
calculate_ISRate, *4*, *6*, *8*, *10*, 10, *14*, *19*, *21*, *23, 24*, *26*, *29*
calculate_ISRatio, *4*, *6*, *8*, *10*, *12*, 13, *19*, *21*, *23, 24*, *26*, *29*

DSR_data, 15

esp2013, 16

LE_data, 17

phe_dsr, *4*, *6*, *8*, *10*, *12*, *14*, *19*, *21*, *23, 24*, *26*, *29*
phe_life_expectancy, *4*, *6*, *8*, *10*, *12*, *14*, 17, *21*, *23, 24*, *26*, *29*
phe_mean, *4*, *6*, *8*, *10*, *12*, *14*, *19*, 20, *23, 24*, *26*, *29*
phe_proportion, *4*, *6*, *8*, *10*, *12*, *14*, *19*, *21*, 22, *24*, *26*, *29*
phe_quantile, *4*, *6*, *8*, *10*, *12*, *14*, *19*, *21*, *23*, 23, *26*, *29*
phe_rate, *4*, *6*, *8*, *10*, *12*, *14*, *19*, *21*, *23, 24*, 25, *29*
phe_sii, *4*, *6*, *8*, *10*, *12*, *14*, *19*, *21*, *23, 24*, *26*, 26
prevalence_data, 30