Package: sleacr (via r-universe)

August 31, 2024

Type Package

Title Simplified Lot Quality Assurance Sampling Evaluation of Access and Coverage (SLEAC) Tools in R

Version 0.0.0.9000

Description In the recent past, measurement of coverage has been mainly through two-stage cluster sampled surveys either as part of a nutrition assessment or through a specific coverage survey known as Centric Systematic Area Sampling (CSAS). However, such methods are resource intensive and often only used for final programme evaluation meaning results arrive too late for programme adaptation. SLEAC, which stands for Simplified Lot Quality Assurance Sampling Evaluation of Access and Coverage, is a low resource method designed specifically to address this limitation and is used regularly for monitoring, planning and importantly, timely improvement to programme quality, both for agency and Ministry of Health (MoH) led programmes. SLEAC is designed to complement the Semi-quantitative Evaluation of Access and Coverage (SQUEAC) method. This package provides functions for use in conducting a SLEAC assessment.

License GPL-3

Depends R (>= 2.10)

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

Encoding UTF-8

Language en-GB

LazyData true

RoxygenNote 7.3.1

Roxygen list(markdown = TRUE)

URL https://nutriverse.io/sleacr/,https://github.com/nutriverse/sleacr

BugReports https://github.com/nutriverse/sleacr/issues

Config/testthat/edition 3

VignetteBuilder knitr

Repository https://nutriverse.r-universe.dev

RemoteUrl https://github.com/nutriverse/sleacr

RemoteRef HEAD

RemoteSha c5a55c02d7c0777e4890c150957cbb1ded807773

Contents

`classify_coverage` *Classify coverage results*

Description

Classify coverage results

Usage

```
classify_coverage(n_in, n_total, standard = c(0.2, 0.5))
```

Arguments

`n_in`	Number (integer) of cases found in the programme
`n_total`	Number (integer) of children under 5 years sampled
`standard`	Decision rule standard/s. Should be between 0 and 1. At least one standard should be provided for a two-tier coverage classifier. Two standards should be provided for a three-tier coverage classifier. Default is a three-tier classifier with rule set at 0.2 and 0.5.

Value

A character value or vector indicating coverage classification. If `standard` is a single value, returns **"Satisfactory"** if coverage is above `standard` and **"Not satisfactory"** if coverage is below or equal to `standard`. If `standard` is two values, returns **"Low"** if coverage is below or equal to lower standard, **"High"** if coverage is above the higher standard, and **"Moderate"** for all other coverage values.

Author(s)

Ernest Guevarra

Examples

```
classify_coverage(n_in = 6, n_total = 40)
with(survey_data,
  classify_coverage(n_in = in_cases, n_total = n)
)
```

`get_binom_hypergeom` *Calculate the binomial coefficient "n-choose-k"*

Description

Calculate the binomial coefficient "n-choose-k"

Usage

```
get_binom_hypergeom(n, k)
```

Arguments

`n`	Total population
`k`	Number of sample drawn from total population

Value

A numeric vector of binomial probability

Examples

```
get_binom_hypergeom(n = 600, k = 40)
```

get_class_prob *Function to produce misclassification probabilities*

Description

Function to produce misclassification probabilities

Usage

```
get_class_prob(x)
```

Arguments

x	Simulated results data produced by test_lqas_classifier()

Value

A list of LQAS misclassification probabilities results

Examples

```
sim <- test_lqas_classifier(replicates = 5, runs = 5,
                            pop = 10000, n = 40,
                            d.lower = 60, d.upper = 90)
get_class_prob(x = sim)
```

get_d *Calculate decision rule for a specified sample size and lower and upper triage thresholds*

Description

Calculate decision rule for a specified sample size and lower and upper triage thresholds

Usage

```
get_d(N, n, dLower, dUpper, alpha = 0.1, beta = 0.1)
```

Arguments

N	Total population size of cases in the specified survey area
n	Sample size
dLower	Lower triage threshold. Values from 0 to 1.
dUpper	Upper triage threshold. Values from 0 to 1.
alpha	Maximum tolerable alpha error. Values from 0 to 1. Default is 0.1
beta	Maximum tolerable beta error. Values from 0 to 1. Default is 0.1

Value

A list of values providing the LQAS sampling plan for the specified parameters. The list includes sample size, decision rule, alpha error and beta error for the specified classification scheme

Examples

```
get_d(N = 600, n = 40, dLower = 0.7, dUpper = 0.9)
```

---

get_hypergeom *Calculate hypergeometric probability*

Description

Calculate hypergeometric probability

Usage

```
get_hypergeom(k, m, n, N)
```

Arguments

k	Number of cases in the sample
m	Number of cases in the population
n	Sample size
N	Population size

Value

A numeric value of hypergeometric probability given specified parameters

Examples

```
get_hypergeom(k = 5, m = 600, n = 25, N = 10000)
```

get_hypergeom_cumulative

Calculate cumulative hypergeometric probabilities

Description

Calculate cumulative hypergeometric probabilities

Usage

```
get_hypergeom_cumulative(k, m, n, N, tail = "lower")
```

Arguments

k	Number of cases in the sample
m	Number of cases in the population
n	Sample size
N	Population size
tail	A character vector indicating "lower" (default) or "upper" tail

Examples

```
get_hypergeom_cumulative(k = 5, m = 600, n = 25, N = 10000)
```

get_n

Calculate sample size of number of cases to be found to assess coverage

Description

Calculate sample size of number of cases to be found to assess coverage

Usage

```
get_n(N, dLower, dUpper, alpha = 0.1, beta = 0.1)
```

Arguments

N	Total population size of cases in the specified survey area
dLower	Lower triage threshold. Values from 0 to 1.
dUpper	Upper triage threshold. Values from 0 to 1.
alpha	Maximum tolerable alpha error. Values from 0 to 1. Default is 0.1
beta	Maximum tolerable beta error. Values from 0 to 1. Default is 0.1

Value

A list of values providing the LQAS sampling plan for the specified parameters. The list includes sample size, decision rule, alpha error and beta error for the specified classification scheme

Examples

```
get_n(N = 600, dLower = 0.7, dUpper = 0.9)
```

`get_n_cases` *Calculate estimated number of cases for a condition affecting children under 5 years old in a specified survey area*

Description

Calculate estimated number of cases for a condition affecting children under 5 years old in a specified survey area

Usage

```
get_n_cases(N, u5, p)
```

Arguments

`N`	Population for all ages in the specified survey area
`u5`	Proportion (value from 0 to 1) of population that are aged 6-59 months
`p`	Prevalence of condition that is to be assessed

Value

Numeric value of the estimated number of cases in the specified survey area

Examples

```
## Calculate number of SAM cases in a population of 100000 persons of all
## ages with an under-5 population of 17% and a prevalence of 2%
get_n_cases(N = 100000, u5 = 0.17, p = 0.02)
```

get_n_clusters *Calculate number of clusters to sample to reach target sample size*

Description

Calculate number of clusters to sample to reach target sample size

Usage

```
get_n_clusters(n, N, u5, p)
```

Arguments

n	Target sample size of cases for the coverage survey
N	Average cluster population for all ages in the specified survey area
u5	Proportion (value from 0 to 1) of population that are aged 6-59 months
p	Prevalence of condition that is to be assessed

Value

Numeric value of the estimated number of clusters to sample to reach target sample size

Examples

```
## Calculate number of villages to sample given an average village population
## of 600 persons of all ages with an under-5 population of 17% and a
## prevalence of SAM of 2% if the target sample size is 40
get_n_clusters(n = 40, N = 600, u5 = 0.17, p = 0.02)
```

get_sampling_interval *Select sampling clusters using systematic sampling*

Description

Select sampling clusters using systematic sampling

Usage

```
get_sampling_interval(N_clusters, n_clusters)

select_random_start(interval)

select_sampling_clusters(N_clusters, n_clusters)

create_sampling_list(cluster_list, n_clusters)
```

Arguments

N_clusters	Total number of clusters in survey area
n_clusters	Number of sampling clusters to be selected
interval	Sampling interval usually calculated using get_sampling_interval()
cluster_list	A data.frame containing at least the name or any other identifier for the entire set of clusters to sample from.

Value

A numeric value for get_sampling_interval() and select_random_start(); An integer vector for select_sampling_clusters() giving the row index for selected clusters; A data.frame for create_sampling_list() which is a subset of cluster_list

Examples

```
get_sampling_interval(N_clusters = 211, n_clusters = 35)
interval <- get_sampling_interval(N_clusters = 211, n_clusters = 35)
select_random_start(interval)
select_sampling_clusters(N_clusters = 211, n_clusters = 35)
create_sampling_list(cluster_list = village_list, n_clusters = 70)
```

---

make_data *Function to simulate survey data of covered/cases and non-covered/non-cases given specified parameters*

Description

Function to simulate survey data of covered/cases and non-covered/non-cases given specified parameters

Usage

```
make_data(proportion, pop)
```

Arguments

proportion	A numeric value of a coverage proportion to simulate on
pop	Population size from which simulated coverage survey data is to be taken from

Value

A numeric vector of cases and non-cases (1s and 0s)

Examples

```
make_data(proportion = 0.3, pop = 10000)
```

plot.lqasSim — plot *helper function for* test_lqas_classifier() *function*

Description

plot helper function for test_lqas_classifier() function

Usage

```
## S3 method for class 'lqasSim'
plot(x, ...)
```

Arguments

x	An object of class lqasSim produced by test_lqas_classifier() function
...	Additional plot parameters

Value

An LQAS probability of classification plot

Examples

```
x <- test_lqas_classifier(replicates = 5, runs = 5,
                          pop = 10000, n = 40, d.lower = 60, d.upper = 90)
plot(x)
```

print.lqasClass — print *helper function for* get_class_prob() *function*

Description

print helper function for get_class_prob() function

Usage

```
## S3 method for class 'lqasClass'
print(x, ...)
```

Arguments

x	An object resulting from applying the get_class_prob() function.
...	Additional print parameters

Value

Printed output of `get_class_prob()` function

Examples

```
sim <- test_lqas_classifier(replicates = 5, runs = 5,
                            pop = 10000, n = 40,
                            d.lower = 60, d.upper = 90)
x <- get_class_prob(x = sim)
print(x)
```

---

run_lqas *Function to perform LQAS based on data based on specified decision rules*

Description

Function to perform LQAS based on data based on specified decision rules

Usage

```
run_lqas(data, n, d.lower, d.upper)
```

Arguments

`data`	A vector of simulated data produced by `make_data()`
`n`	Sample size of actual or test coverage data
`d.lower`	A numeric value for the lower classification threshold
`d.upper`	A numeric value for the upper classification threshold

Value

A list of coverage proportions and LQAS outcomes

Examples

```
run_lqas(data = make_data(proportion = 0.3, pop = 10000),
         n = 40, d.lower = 60, d.upper = 90)
```

`simulate_lqas` *Function to perform a series of LQAS analysis on simulated coverage survey data*

Description

Function to perform a series of LQAS analysis on simulated coverage survey data

Usage

```
simulate_lqas(
  runs = 50,
  pop = NULL,
  n,
  d.lower,
  d.upper,
  p.lower = 0,
  p.upper = 100,
  fine = 1,
  progress = TRUE
)
```

Arguments

`runs`	Number of simulation runs to perform per coverage proportion. Default is 50 runs
`pop`	Population size from which simulated coverage survey data is to be taken from
`n`	Sample size of actual or test coverage data
`d.lower`	A numeric value for the lower classification threshold
`d.upper`	A numeric value for the upper classification threshold
`p.lower`	Starting proportion for simulations. Default is 0
`p.upper`	Ending proportion for simulations. Default is 100
`fine`	Granularity of simulated proportions; Default to 1
`progress`	Logical. Should simulation progress be shown? Default is TRUE

Value

A data.frame of coverage proportions and LQAS outcomes

Examples

```
simulate_lqas(runs = 10, pop = 10000, n = 40, d.lower = 60, d.upper = 90)
```

survey_data *SLEAC survey data from Sierra Leone*

Description

SLEAC survey data from Sierra Leone

Usage

```
survey_data
```

Format

A tibble with 14 rows and 6 columns:

`country` Country

`province` Province

`district` District

`in_cases` Cases found who are in the programme

`out_cases` Cases found who are not in the programme

`n` Total number of under 5 children sampled

Source

Ministry of Health, Sierra Leone

Examples

```
survey_data
```

test_lqas_classifier *Function to test performance of LQAS classifier*

Description

Function to test performance of LQAS classifier

Usage

```
test_lqas_classifier(
  replicates = 20,
  runs = 50,
  pop = NULL,
  n = NULL,
  d.lower = NULL,
  d.upper = NULL,
  p.lower = 0,
  p.upper = 100,
  fine = 1,
  progress = TRUE
)
```

Arguments

replicates	Number of replicate LQAS simulations to perform. Default is set to 20 replicates
runs	Number of simulation runs to perform per coverage proportion. Default is 50 runs
pop	Population size from which simulated coverage survey data is to be taken from
n	Sample size of actual or test coverage data
d.lower	A numeric value for the lower classification threshold
d.upper	A numeric value for the upper classification threshold
p.lower	Starting proportion for simulations. Default is 0
p.upper	Ending proportion for simulations. Default is 100
fine	Granularity of simulated proportions; Defaul to 1
progress	Logical. Should simulation progress be shown? Default is TRUE

Value

A sleac object

Examples

```
test_lqas_classifier(replicates = 5, runs = 5,
                     pop = 10000, n = 40, d.lower = 60, d.upper = 90)
```

`village_list` *List of villages in Bo District, Sierra Leone*

Description

List of villages in Bo District, Sierra Leone

Usage

```
village_list
```

Format

A tibble with 1001 rows and 4 columns:

`id` Unique identifier

`chiefdom` Chiefdom

`section` Section

`village` Village

Source

Ministry of Health, Sierra Leone

Examples

```
village_list
```

Index