Package 'stratallo'

November 27, 2023

Title Optimum Sample Allocation in Stratified Sampling

Version 2.2.1

Description Functions in this package provide solution to classical problem in survey methodology - an optimum sample allocation in stratified sampling. In this context, the optimum allocation is in the classical Tschuprow-Neyman's sense and it satisfies additional lower or upper bounds restrictions imposed on sample sizes in strata. There are few different algorithms available to use, and one them is based on popular sample allocation method that applies Neyman allocation to recursively reduced set of strata.
This package also provides the function that computes a solution to the minimum cost allocation problem, which is a minor modification of the classical optimum sample allocation. This problem lies in the determination of a vector of strata sample sizes that minimizes total cost of the survey, under assumed fixed level of the stratified estimator's variance. As in the case of the classical optimum allocation, the problem of minimum cost allocation can be complemented by imposing upper-bounds constraints on sample sizes in strata.

License GPL-2

URL https://github.com/wwojciech/stratallo

BugReports https://github.com/wwojciech/stratallo/issues

Copyright Wojciech Wójciak

Language en-US

Encoding UTF-8

LazyData true

Imports checkmate, lifecycle

RoxygenNote 7.2.3

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

Config/testthat/edition 3

Collate 'helpers.R' 'algorithms_1sided.R' 'algorithms_2sided.R' 'algorithms_of_other_authors.R' 'data.R' 'opt.R' 'stratallo-package.R'

VignetteBuilder knitr

Depends R (>= 2.10)

NeedsCompilation no

Author Wojciech Wójciak [aut, cre],
Jacek Wesołowski [sad],
Robert Wieczorkowski [ctb]

Maintainer Wojciech Wójciak <wojciech.wojciak@gmail.com>

Repository CRAN

Date/Publication 2023-11-26 23:20:02 UTC

R topics documented:

`stratallo-package` *Functions for Optimum Sample Allocation in Stratified Sampling*

Description

Optimum Sample Allocation in Stratified Sampling

Author(s)

Wojciech Wójciak <wojciech.wojciak@gmail.com>

References

Stenger, H., Gabler, S. (2005). Combining random sampling and census strategies - Justification of inclusion probabilities equal to 1. *Metrika*, 61(2), pp. 137–156. doi:10.1007/s001840400328

Särndal, C.-E., Swensson, B. and Wretman, J. (1992). *Model Assisted Survey Sampling*, Springer, New York.

Wesołowski, J., Wieczorkowski, R., Wójciak, W. (2021). Optimality of the Recursive Neyman Allocation. *Journal of Survey Statistics and Methodology*, 10(5), pp. 1263–1275. doi:10.1093/jssam/smab018, doi:10.48550/arXiv.2105.14486

Wesołowski, J., Wieczorkowski, R., Wójciak, W. (2023). R Package stratallo - source code (Version 2.2.0). https://github.com/wwojciech/stratallo

Wesołowski, J., Wieczorkowski, R., Wójciak, W. (2023). Numerical Performance of the RNABOX Algorithm (Version 1.0.1). https://github.com/rwieczor/recursive_Neyman_rnabox

Wójciak, W. (2023). Another Solution of Some Optimum Allocation Problem. *Statistics in Transition new series*, 24(5) (in press). https://arxiv.org/abs/2204.04035

Wójciak, W. (2019). Optimal Allocation in Stratified Sampling Schemes. *MSc Thesis*, Warsaw University of Technology, Warsaw, Poland. http://home.elka.pw.edu.pl/~wwojciak/msc_optimal_allocation.pdf

---

`asummary` *Summarizing the Allocation*

Description

[Stable]

A helper function that returns a simple `data.frame` with summary of the allocation as returned by the `opt()` or `optcost()`. See the illustrate example below.

Usage

```
asummary(x, A, m = NULL, M = NULL)
```

Arguments

`x`	(`numeric`) sample allocations $x_1, \ldots, x_H$ in strata.
`A`	(`numeric`) population constants $A_1, \ldots, A_H$.

`m` (`numeric` or `NULL`)
lower bounds $m_1, \ldots, m_H$, optionally imposed on sample sizes in strata.

`M` (`numeric` or `NULL`)
upper bounds $M_1, \ldots, M_H$, optionally imposed on sample sizes in strata.

Value

A `data.frame` with as many rows as number of strata H + 1, and up to 7 variables. A single row corresponds to a given stratum $h \in \{1, \ldots, H\}$, whilst the last row contains sums of all of the numerical values from the above rows (wherever feasible). Summary table has the following columns (* indicates that the column may not be present):

A population constant A_h

m* lower bound imposed on sample size in stratum

M* upper bound imposed on sample size in stratum

allocation sample size for a given stratum

take_min* indication whether the allocation is of `take-min` type, i.e. $x_h = m_h$

take_max* indication whether the allocation is of `take-max` type, i.e. $x_h = M_h$

take_Neyman indication whether the allocation is of `take-Neyman` type, i.e. $m_h < x_h < M_h$

See Also

`opt()`, `optcost()`.

Examples

```
A <- c(3000, 4000, 5000, 2000)
m <- c(100, 90, 70, 80)
M <- c(200, 150, 300, 210)

xopt_1 <- opt(n = 400, A, m)
asummary(xopt_1, A, m)

xopt_2 <- opt(n = 540, A, m, M)
asummary(xopt_2, A, m, M)
```

`CapacityScaling` *Integer-valued Optimal Univariate Allocation Under Constraints for Stratified Sampling*

Description

[Experimental]

Better algorithm from paper Friedrich et al. (2015) for integer-valued optimal allocation in stratified sampling.

Usage

```
CapacityScaling(n, Ah, mh = rep(1, length(Ah)), Mh = rep(Inf, length(Ah)))

CapacityScaling2(
  v0,
  Nh,
  Sh,
  mh = rep(1, length(Nh)),
  Mh = rep(Inf, length(Nh))
)
```

Arguments

n	• target sample size for allocation.
Ah	• population strata sizes * standard deviations of a given variable in strata.
mh	• lower constraints for sample sizes in strata.
Mh	• upper constraints for sample sizes in strata.
v0	• upper limit for value of variance which must be attained for computed optimal allocation.
Nh	• population strata sizes.
Sh	• standard deviations of a given variable in strata.

Value

A vector of optimal allocation sizes.

Functions

- `CapacityScaling2()`:

References

Friedrich, U., Münnich, R., de Vries, S. and Wagner, M. (2015) Fast integer-valued algorithms for optimal allocations under constraints in stratified sampling, *Computational Statistics and Data Analysis*, 92, pp. 1–12. https://www.sciencedirect.com/science/article/pii/S0167947315001413

`fpia` *Optimal Univariate Allocation Under Constraints for Stratified Sampling*

Description

[Experimental]

Algorithm for optimal allocation in stratified sampling with lower and upper constraints based on fixed point iteration.

Usage

```
fpia(
  n,
  Ah,
  mh = NULL,
  Mh = NULL,
  lambda0 = NULL,
  maxiter = 100,
  tol = .Machine$double.eps * 1000
)

fpia2(v0, Nh, Sh, mh = NULL, Mh = NULL, lambda0 = NULL, maxiter = 100)
```

Arguments

`n`	• target sample size for allocation.
`Ah`	• population strata sizes * standard deviations of a given variable in strata.
`mh`	• lower constraints for sample sizes in strata.
`Mh`	• upper constraints for sample sizes in strata.
`lambda0`	• initial parameter 'lambda' (optional).
`maxiter`	• maximal number of iterations for algorithm.
`tol`	• the desired accuracy (convergence tolerance).
`v0`	• upper limit for value of variance which must be attained for computed optimal allocation.
`Nh`	• population strata sizes.
`Sh`	• standard deviations of a given variable in strata.

Value

A vector of optimal allocation sizes, and number of iterations.

Functions

- `fpia2()`:

References

Münnich, R. T., Sachs, E.W. and Wagner, M. (2012) Numerical solution of optimal allocation problems in stratified sampling under box constraints, *AStA Advances in Statistical Analysis*, 96(3), pp. 435-450. doi:10.1007/s101820110176z

opt *Optimum Sample Allocation in Stratified Sampling*

Description

[Stable]

A classical problem in survey methodology in stratified sampling is optimum sample allocation. This problem is formulated as determination of strata sample sizes that minimize the variance of the *stratified π estimator* of the population total (or mean) of a given study variable, under certain constraints on sample sizes in strata.

The `opt()` user function solves the following optimum sample allocation problem, formulated below in the language of mathematical optimization.

Minimize

$$f(x_1, \ldots, x_H) = \sum_{h=1}^{H} \frac{A_h^2}{x_h}$$

subject to

$$\sum_{h=1}^{H} x_h = n$$

$$m_h \leq x_h \leq M_h, \quad h = 1, \ldots, H,$$

where $n > 0$, $A_h > 0$, $m_h > 0$, $M_h > 0$, such that $m_h < M_h$, $h = 1, \ldots, H$, and $\sum_{h=1}^{H} m_h \leq n \leq \sum_{h=1}^{H} M_h$, are given numbers. The minimization is on $\mathbb{R}_+^H$.

The inequality constraints are optional and user can choose whether and how they are to be added to the optimization problem. This is achieved by the proper use of `m` and `M` arguments of this function, according to the following rules:

- no inequality constraints imposed: both `m` and `M` must be both set to `NULL` (default).
- one-sided lower bounds m_h, $h = 1, \ldots, H$, imposed: lower bounds are specified with `m`, while `M` is set to `NULL`.
- one-sided upper bounds M_h, $h = 1, \ldots, H$, imposed: upper bounds are specified with `M`, while `m` is set to `NULL`.
- box-constraints imposed: lower and upper bounds must be specified with `m` and `M`, respectively.

Usage

```
opt(n, A, m = NULL, M = NULL, M_algorithm = "rna")
```

Arguments

`n`	(`number`) total sample size. A strictly positive scalar. If `bounds1` is not `NULL`, it is then required that `n >= sum(bounds1)` (given that `bounds1` are treated as lower bounds) or `n <= sum(bounds1)` (given that `bounds1` are treated as upper bounds). If `bounds2` is not `NULL`, it is then required that `n >= sum(bounds2)` (given that

bounds2 are treated as lower bounds) or n <= sum(bounds2) (given that bounds2 are treated as upper bounds).

A (numeric)
population constants $A_1, \ldots, A_H$. Strictly positive numbers.

m (numeric or NULL)
lower bounds $m_1, \ldots, m_H$, optionally imposed on sample sizes in strata. If no lower bounds should be imposed, then m must be set to NULL. If M is not NULL, it is then required that m < M.

M (numeric or NULL)
upper bounds $M_1, \ldots, M_H$, optionally imposed on sample sizes in strata. If no upper bounds should be imposed, then M must be set to NULL. If m is not NULL, it is then required that m < M.

M_algorithm (string)
the name of the underlying algorithm to be used for computing sample allocation under one-sided upper-bounds constraints. It must be one of the following: rna (default), sga, sgaplus, coma. This parameter is used only in case when m argument is NULL and M is not NULL and number of strata $H > 1$ and n < sum(M).

Details

The opt() function makes use of several allocation algorithms, depending on which of the inequality constraints should be taken into account in the optimization problem. Each algorithm is implemented in a separate R function that in general should not be used directly by the end user. The following is the list with the algorithms that are used along with the name of the function that implements a given algorithm. See the description of a specific function to find out more about the corresponding algorithm.

- one-sided lower-bounds m_h, $h = 1, \ldots, H$:
  - LRNA - rna()
- one-sided upper-bounds M_h, $h = 1, \ldots, H$:
  - RNA - rna()
  - SGA - sga()
  - SGAPLUS - sgaplus()
  - COMA - coma()
- box constraints m_h, M_h, $h = 1, \ldots, H$:
  - RNABOX - rnabox()

Value

Numeric vector with optimal sample allocations in strata.

Note

If no inequality constraints are added, the allocation is given by the Neyman allocation as:

$$x_h = A_h \frac{n}{\sum_{i=1}^{H} A_i}, \quad h = 1, \ldots, H.$$

For *stratified π estimator* of the population total with *stratified simple random sampling without replacement* design in use, the parameters of the objective function f are:

$$A_h = N_h S_h, \quad h = 1, \ldots, H,$$

where N_h is the size of stratum h and S_h denotes standard deviation of a given study variable in stratum h.

References

Särndal, C.-E., Swensson, B. and Wretman, J. (1992). *Model Assisted Survey Sampling*, Springer, New York.

See Also

optcost(), rna(), sga(), sgaplus(), coma(), rnabox().

Examples

```
A <- c(3000, 4000, 5000, 2000)
m <- c(100, 90, 70, 50)
M <- c(300, 400, 200, 90)

# One-sided lower bounds.
opt(n = 340, A = A, m = m)
opt(n = 400, A = A, m = m)
opt(n = 700, A = A, m = m)

# One-sided upper bounds.
opt(n = 190, A = A, M = M)
opt(n = 700, A = A, M = M)

# Box-constraints.
opt(n = 340, A = A, m = m, M = M)
opt(n = 500, A = A, m = m, M = M)
xopt <- opt(n = 800, A = A, m = m, M = M)
xopt
var_st(x = xopt, A = A, A0 = 45000) # Value of the variance for allocation xopt.

# Execution-time comparisons of different algorithms with microbenchmark R package.
## Not run:
N <- pop969[, "N"]
S <- pop969[, "S"]
A <- N * S
nfrac <- c(0.005, seq(0.05, 0.95, 0.05))
n <- setNames(as.integer(nfrac * sum(N)), nfrac)
lapply(
  n,
  function(ni) {
    microbenchmark::microbenchmark(
      RNA = opt(ni, A, M = N, M_algorithm = "rna"),
      SGA = opt(ni, A, M = N, M_algorithm = "sga"),
```

```
      SGAPLUS = opt(ni, A, M = N, M_algorithm = "sgaplus"),
      COMA = opt(ni, A, M = N, M_algorithm = "coma"),
      times = 200,
      unit = "us"
    )
  }
)

## End(Not run)
```

---

optcost *Minimum Cost Allocation in Stratified Sampling*

Description

[Stable]

Function that determines fixed strata sample sizes that minimize total cost of the survey, under assumed level of the variance of the stratified estimator and under optional one-sided upper bounds imposed on strata sample sizes. Namely, the following optimization problem, formulated below in the language of mathematical optimization, is solved by `optcost()` function.

Minimize

$$c(x_1, \ldots, x_H) = \sum_{h=1}^{H} c_h x_h$$

subject to

$$\sum_{h=1}^{H} \frac{A_h^2}{x_h} - A_0 = V$$

$$x_h \leq M_h, \quad h = 1, \ldots, H,$$

where A_0, $A_h > 0$, $c_h > 0$, $M_h > 0$, $h = 1, \ldots, H$, and $V > \sum_{h=1}^{H} \frac{A_h^2}{M_h} - A_0$ are given numbers. The minimization is on $\mathbb{R}_+^H$. The upper-bounds constraints $x_h \leq M_h$, $h = 1, \ldots, H$, are optional and can be skipped. In such a case, it is only required that $V > 0$.

Usage

```
optcost(V, A, A0, M = NULL, unit_costs = 1)
```

Arguments

`V`	(`number`) parameter V of the equality constraint. A strictly positive scalar. If `M` is not `NULL`, it is then required that `V >= sum(A^2/M) - A0`.
`A`	(`numeric`) population constants $A_1, \ldots, A_H$. Strictly positive numbers.
`A0`	(`number`) population constant A_0.

`M` (`numeric` or `NULL`)
upper bounds $M_1, \ldots, M_H$, optionally imposed on sample sizes in strata. If no upper bounds should be imposed, then `M` must be set to `NULL`.

`unit_costs` (`numeric`)
costs $c_1, \ldots, c_H$, of surveying one element in stratum. A strictly positive numbers. Can be also of length 1, if all unit costs are the same for all strata. In this case, the elements will be recycled to the length of `bounds`.

Details

The algorithm that is used by `optcost()` is the `LRNA` and it is described in Wójciak (2023). The allocation computed is valid for all stratified sampling schemes for which the variance of the stratified estimator is of the form:

$$\sum_{h=1}^{H} \frac{A_h^2}{x_h} - A_0,$$

where H denotes total number of strata, $x_1, \ldots, x_H$ are strata sample sizes and A_0, $A_h > 0$, $h = 1, \ldots, H$, do not depend on x_h, $h = 1, \ldots, H$.

Value

Numeric vector with optimal sample allocations in strata.

Note

For *stratified π estimator* of the population total and for *stratified simple random sampling without replacement* design, the population parameters are as follows:

$$A_h = N_h S_h, \quad h = 1, \ldots, H,$$

$$A_0 = \sum_{h=1}^{H} N_h S_h^2,$$

where N_h is the size of stratum h and S_h denotes standard deviation of a given study variable in stratum h.

References

Wójciak, W. (2023). Another Solution of Some Optimum Allocation Problem. *Statistics in Transition new series*, 24(5) (in press). https://arxiv.org/abs/2204.04035

See Also

`rna()`, `opt()`.

Examples

```
A <- c(3000, 4000, 5000, 2000)
M <- c(100, 90, 70, 80)
xopt <- optcost(1017579, A = A, A0 = 579, M = M)
xopt
```

opt_1sided *Algorithms for Optimum Sample Allocation Under One-Sided Bounds*

Description

[Stable]

Functions that implement selected optimal allocation algorithms that compute a solution to the optimal allocation problem defined in the language of mathematical optimization as follows.

Minimize

$$f(x_1, \ldots, x_H) = \sum_{h=1}^{H} \frac{A_h^2}{x_h}$$

subject to

$$\sum_{h=1}^{H} c_h x_h = c$$

and either

$$x_h \leq M_h, \quad h = 1, \ldots, H$$

or

$$x_h \geq m_h, \quad h = 1, \ldots, H,$$

where $c > 0$, $c_h > 0$, $A_h > 0$, $m_h > 0$, $M_h > 0$, $h = 1, \ldots, H$, are given numbers. The minimization is on $\mathbb{R}_+^H$.

The inequality constraints are optional and user can choose whether and how they are to be added to the optimization problem. If one-sided lower bounds m_h, $h = 1, \ldots, H$, must be imposed, it is then required that $c \geq \sum_{h=1}^{H} c_h m_h$. If one-sided upper bounds M_h, $h = 1, \ldots, H$, must be imposed, it is then required that $0 < c \leq \sum_{h=1}^{H} c_h M_h$. Lower bounds can be specified instead of the upper bounds only in case of the *LRNA* algorithm. All other algorithms allow only for specification of the upper bounds. For the sake of clarity, we emphasize that in the optimization problem consider here, the lower and upper bounds cannot be imposed jointly.

Costs c_h, $h = 1, \ldots, H$, of surveying one element in stratum, can be specified by the user only in case of the *RNA* and *LRNA* algorithms. For remaining algorithms, these costs are fixed at 1, i.e. $c_h = 1$, $h = 1, \ldots, H$.

The following is the list of all the algorithms available to use along with the name of the function that implements a given algorithm. See the description of a specific function to find out more about the corresponding algorithm.

- *RNA* - `rna()`
- *LRNA*- `rna()`
- *SGA*- `sga()`
- *SGAPLUS* - `sgaplus()`
- *COMA* - `coma()`

Functions in this family should not be called directly by the user. Use `opt()` or `optcost()` instead.

Usage

```
rna(
  total_cost,
  A,
  bounds = NULL,
  unit_costs = 1,
  check_violations = .Primitive(">="),
  details = FALSE
)

sga(total_cost, A, M)

sgaplus(total_cost, A, M)

coma(total_cost, A, M)
```

Arguments

`total_cost`	(`number`) total cost c of the survey. A strictly positive scalar.
`A`	(`numeric`) population constants $A_1, \ldots, A_H$. Strictly positive numbers.
`bounds`	(`numeric` or `NULL`) optional lower bounds $m_1, \ldots, m_H$, or upper bounds $M_1, \ldots, M_H$, or `NULL` to indicate that there is no inequality constraints in the optimization problem considered. If not `NULL`, the `bounds` is to be treated either as: • lower bounds, if `check_violations = .Primitive("<=")`. In this case, it is required that `total_cost >= sum(unit_costs * bounds)`, or • upper bounds, if `check_violations = .Primitive(">=")`. In this case, it is required that `total_cost <= sum(unit_costs * bounds)`.
`unit_costs`	(`numeric`) costs $c_1, \ldots, c_H$, of surveying one element in stratum. A strictly positive numbers. Can be also of length 1, if all unit costs are the same for all strata. In this case, the elements will be recycled to the length of `bounds`.
`check_violations`	(`function`) 2-arguments binary operator function that allows the comparison of values in atomic vectors. It must either be set to `.Primitive("<=")` or `.Primitive(">=")`. The first of these choices causes that `bounds` are treated as lower bounds and then `rna()` function performs the *LRNA* algorithm. The latter option causes that `bounds` are treated as upper bounds, and then `rna()` function performs the *RNA* algorithm. This argument is ignored when `bounds` is set to `NULL`.
`details`	(`flag`) should detailed information about strata assignments (either to take-Neyman or take-bound), values of set function s and number of iterations be added to the output?

`M` (`numeric` or `NULL`)
upper bounds $M_1, \ldots, M_H$, optionally imposed on sample sizes in strata. If no upper bounds should be imposed, then `M` must be set to `NULL`. Otherwise, it is required that `total_cost <= sum(unit_costs * M)`. Strictly positive numbers.

Value

Numeric vector with optimal sample allocations in strata. In case of the `rna()` only, it can also be a `list` with optimal sample allocations and strata assignments (either to take-Neyman or take-bound).

Functions

- `rna()`: *Recursive Neyman Algorithm* (*RNA*) and its twin version, *Lower Recursive Neyman Algorithm* (*LRNA*) dedicated to the allocation problem with one-sided lower-bounds constraints. The *RNA* is described in Wesołowski et al. (2021), while *LRNA* is introduced in Wójciak (2023).
- `sga()`: Stenger-Gabler type algorithm *SGA*, described in Wesołowski et al. (2021) and in Stenger and Gabler (2005). This algorithm solves the problem with one-sided upper-bounds constraints. It also assumes unit costs are constant and equal to 1, i.e. $c_h = 1,\ h = 1, \ldots, H$.
- `sgaplus()`: modified Stenger-Gabler type algorithm, described in Wójciak (2019) as *Sequential Allocation (version 1)* algorithm. This algorithm solves the problem with one-sided upper-bounds constraints. It also assumes unit costs are constant and equal to 1, i.e. $c_h = 1,\ h = 1, \ldots, H$.
- `coma()`: *Change of Monotonicity Algorithm* (*COMA*), described in Wesołowski et al. (2021). This algorithm solves the problem with one-sided upper-bounds constraints. It also assumes unit costs are constant and equal to 1, i.e. $c_h = 1,\ h = 1, \ldots, H$.

Note

If no inequality constraints are added, the allocation is given by the Neyman allocation as:

$$x_h = \frac{A_h}{\sqrt{c_h}} \frac{n}{\sum_{i=1}^{H} A_i \sqrt{c_i}}, \quad h = 1, \ldots, H.$$

For *stratified π estimator* of the population total with *stratified simple random sampling without replacement* design in use, the parameters of the objective function f are:

$$A_h = N_h S_h, \quad h = 1, \ldots, H,$$

where N_h is the size of stratum h and S_h denotes standard deviation of a given study variable in stratum h.

References

Wójciak, W. (2023). Another Solution of Some Optimum Allocation Problem. *Statistics in Transition new series*, 24(5) (in press). https://arxiv.org/abs/2204.04035

Wesołowski, J., Wieczorkowski, R., Wójciak, W. (2021). Optimality of the Recursive Neyman Allocation. *Journal of Survey Statistics and Methodology*, 10(5), pp. 1263–1275. doi:10.1093/jssam/

smab018, doi:10.48550/arXiv.2105.14486

Wójciak, W. (2019). Optimal Allocation in Stratified Sampling Schemes. *MSc Thesis*, Warsaw University of Technology, Warsaw, Poland. http://home.elka.pw.edu.pl/~wwojciak/msc_optimal_allocation.pdf

Stenger, H., Gabler, S. (2005). Combining random sampling and census strategies - Justification of inclusion probabilities equal to 1. *Metrika*, 61(2), pp. 137–156. doi:10.1007/s001840400328

Särndal, C.-E., Swensson, B. and Wretman, J. (1992). *Model Assisted Survey Sampling*, Springer, New York.

See Also

opt(), optcost(), rnabox().

Examples

```
A <- c(3000, 4000, 5000, 2000)
m <- c(50, 40, 10, 30) # lower bounds
M <- c(100, 90, 70, 80) # upper bounds

rna(total_cost = 190, A = A, bounds = M)
rna(total_cost = 190, A = A, bounds = m, check_violations = .Primitive("<="))
sga(total_cost = 190, A = A, M = M)
sgaplus(total_cost = 190, A = A, M = M)
coma(total_cost = 190, A = A, M = M)
```

---

pop10_mM *Example Population with 10 Strata and Lower and Upper Bounds*

Description

A dataset containing the artificial population with 10 strata. Additionally, the lower and upper bounds for samples in strata are specified.

Usage

```
pop10_mM
```

Format

A matrix with 10 rows and 5 variables:

N stratum size

S standard deviation of study variable in stratum

m lower bound for sample size in stratum

M upper bound for sample size in stratum

unit_cost cost of surveying one element in stratum

`pop507` *Example Population with 507 Strata*

Description

A dataset containing the artificial population with 507 strata.

Usage

```
pop507
```

Format

A matrix with 507 rows and 3 variables:

N stratum size

S standard deviation of study variable in stratum

unit_cost cost of surveying one element in stratum

`pop969` *Example Population with 969 Strata*

Description

A dataset containing the artificial population with 969 strata.

Usage

```
pop969
```

Format

A matrix with 969 rows and 3 variables:

N stratum size

S standard deviation of study variable in stratum

unit_cost cost of surveying one element in stratum

ran_round *Random Rounding of Numbers*

Description

[Stable]

A number x is rounded to integer y according to the following rule:

$$y = \lfloor x \rfloor + I(u < (x - \lfloor x \rfloor)),$$

where function $I : \{TRUE, FALSE\} \rightarrow \{0, 1\}$, is defined as:

$$I(x) = \begin{cases} 0, & x \text{ is } FALSE \\ 1, & x \text{ is } TRUE, \end{cases}$$

and u is number that is generated from `Uniform(0, 1)` distribution.

Usage

```
ran_round(x)
```

Arguments

x	(numeric) a numeric vector.

Value

An integer vector.

Examples

```
x <- c(4.5, 4.1, 4.9)
set.seed(5)
ran_round(x) # 5 4 4
set.seed(6)
ran_round(x) # 4 4 5
```

rnabox *Recursive Neyman Algorithm for Optimal Sample Allocation Under Box Constraints*

Description

[Stable]

An internal function that implements the RNABOX algorithm that solves the following optimal allocation problem, formulated below in the language of mathematical optimization.

Minimize

$$f(x_1, \ldots, x_H) = \sum_{h=1}^{H} \frac{A_h^2}{x_h}$$

subject to

$$\sum_{h=1}^{H} x_h = n$$

$$m_h \leq x_h \leq M_h, \quad h = 1, \ldots, H,$$

where $n > 0$, $A_h > 0$, $m_h > 0$, $M_h > 0$, such that $m_h < M_h$, $h = 1, \ldots, H$, and $\sum_{h=1}^{H} m_h \leq n \leq \sum_{h=1}^{H} M_h$, are given numbers. The minimization is on $\mathbb{R}_+^H$. Inequality constraints are optional and can be skipped.

`rnabox()` function should not be called directly by the user. Use `opt()` instead.

Usage

```
rnabox(
  n,
  A,
  bounds1 = NULL,
  bounds2 = NULL,
  check_violations1 = .Primitive(">="),
  check_violations2 = .Primitive("<=")
)
```

Arguments

n	(number) total sample size. A strictly positive scalar. If bounds1 is not NULL, it is then required that n >= sum(bounds1) (given that bounds1 are treated as lower bounds) or n <= sum(bounds1) (given that bounds1 are treated as upper bounds). If bounds2 is not NULL, it is then required that n >= sum(bounds2) (given that bounds2 are treated as lower bounds) or n <= sum(bounds2) (given that bounds2 are treated as upper bounds).
A	(numeric) population constants $A_1, \ldots, A_H$. Strictly positive numbers.

bounds1 (numeric or NULL)
lower bounds $m_1, \ldots, m_H$, or upper bounds $M_1, \ldots, M_H$ optionally imposed on sample sizes in strata. The interpretation of bounds1 depends on the value of check_violations1. If no one-sided bounds 1 should be imposed, then bounds1 must be set to NULL. If bounds2 is not NULL, it is then required that either bounds1 < bounds2 (in case when bounds1 is treated as lower bounds) or bounds1 > bounds2 (in the opposite case).

bounds2 (numeric or NULL)
lower bounds $m_1, \ldots, m_H$, or upper bounds $M_1, \ldots, M_H$ optionally imposed on sample sizes in strata. The interpretation of bounds2 depends on the value of check_violations2. If no one-sided bounds 2 should be imposed, then bounds2 must be set to NULL. If bounds2 is not NULL, it is then required that either bounds1 < bounds2 (in case when bounds1 is treated as lower bounds) or bounds1 > bounds2 (in the opposite case).

check_violations1
(function)
2-arguments binary operator function that allows the comparison of values in atomic vectors. It must either be set to .Primitive("<=") or .Primitive(">="). The first of these choices causes that bounds1 are treated as lower bounds and the rnabox() uses the *LRNA* algorithm as in interim algorithm for the allocation problem with one-sided lower bounds bounds1. The latter option causes that bounds1 are treated as upper bounds and the rnabox() uses the *RNA* algorithm as in interim algorithm for the allocation problem with one-sided upper bounds bounds1. This parameter is correlated with check_violations2. That is, these arguments must be set against each other. check_violations1 is ignored when bounds1 is set to NULL.

check_violations2
(function)
2-arguments binary operator function that allows the comparison of values in atomic vectors. It must either be set to .Primitive("<=") or .Primitive(">="). The first of these choices causes that bounds2 are treated as lower bounds and the rnabox() uses the *LRNA* algorithm as in interim algorithm for the allocation problem with one-sided lower bounds bounds2. The latter option causes that bounds2 are treated as upper bounds and the rnabox() uses the *RNA* algorithm as in interim algorithm for the allocation problem with one-sided upper bounds bounds2. This parameter is correlated with check_violations1. That is, these arguments must be set against each other. check_violations2 is ignored when bounds2 is set to NULL.

Value

Numeric vector with optimal sample allocations in strata.

References

To be added soon.

See Also

opt(), optcost(), sga(), sgaplus(), coma().

Examples

```
N <- c(454, 10, 116, 2500, 2240, 260, 39, 3000, 2500, 400)
S <- c(0.9, 5000, 32, 0.1, 3, 5, 300, 13, 20, 7)
A <- N * S
m <- c(322, 3, 57, 207, 715, 121, 9, 1246, 1095, 294) # lower bounds
M <- N # upper bounds

# Regular allocation.
n <- 6000
opt_regular <- rnabox(n, A, M, m)

# Vertex allocation.
n <- 4076
opt_vertex <- rnabox(n, A, M, m)
```

---

rna_prior *RNA in version that uses prior information about violations*

Description

[Experimental]

This is the version of the RNA that makes use of additional information about strata for which the allocation can possibly be violated. For all other strata allocation will not be violated.

Usage

```
rna_prior(
  total_cost,
  A,
  bounds = NULL,
  check = NULL,
  check_violations = .Primitive(">="),
  details = FALSE
)
```

Arguments

total_cost	(number) total cost c of the survey. A strictly positive scalar.
A	(numeric) population constants $A_1, \ldots, A_H$. Strictly positive numbers.

bounds (numeric or NULL)
optional lower bounds $m_1, \ldots, m_H$, or upper bounds $M_1, \ldots, M_H$, or NULL to indicate that there is no inequality constraints in the optimization problem considered. If not NULL, the bounds is to be treated either as:

- lower bounds, if check_violations = .Primitive("<="). In this case, it is required that total_cost >= sum(unit_costs * bounds),
  or
- upper bounds, if check_violations = .Primitive(">="). In this case, it is required that total_cost <= sum(unit_costs * bounds).

check (integer)
strata indices for which the allocation can possible be violated. For other strata allocation cannot be violated.

check_violations
(function)
2-arguments binary operator function that allows the comparison of values in atomic vectors. It must either be set to .Primitive("<=") or .Primitive(">="). The first of these choices causes that bounds are treated as lower bounds and then rna() function performs the *LRNA* algorithm. The latter option causes that bounds are treated as upper bounds, and then rna() function performs the *RNA* algorithm. This argument is ignored when bounds is set to NULL.

details (flag)
should detailed information about strata assignments (either to take-Neyman or take-bound), values of set function s and number of iterations be added to the output?

Note

this coded was not extensively tested.

---

rna_rec *RNA - Recursive Implementation*

Description

[Experimental]

Usage

```
rna_rec(
  total_cost,
  A,
  bounds = NULL,
  unit_costs = rep(1, length(A)),
  check_violations = .Primitive(">=")
)
```

Arguments

`total_cost` (`number`)
total cost c of the survey. A strictly positive scalar.

`A` (`numeric`)
population constants $A_1, \ldots, A_H$. Strictly positive numbers.

`bounds` (`numeric` or `NULL`)
optional lower bounds $m_1, \ldots, m_H$, or upper bounds $M_1, \ldots, M_H$, or `NULL` to indicate that there is no inequality constraints in the optimization problem considered. If not `NULL`, the `bounds` is to be treated either as:

- lower bounds, if `check_violations = .Primitive("<=")`. In this case, it is required that `total_cost >= sum(unit_costs * bounds)`,
  or
- upper bounds, if `check_violations = .Primitive(">=")`. In this case, it is required that `total_cost <= sum(unit_costs * bounds)`.

`unit_costs` (`numeric`)
costs $c_1, \ldots, c_H$, of surveying one element in stratum. A strictly positive numbers. Can be also of length 1, if all unit costs are the same for all strata. In this case, the elements will be recycled to the length of `bounds`.

`check_violations` (`function`)
2-arguments binary operator function that allows the comparison of values in atomic vectors. It must either be set to `.Primitive("<=")` or `.Primitive(">=")`. The first of these choices causes that `bounds` are treated as lower bounds and then `rna()` function performs the *LRNA* algorithm. The latter option causes that `bounds` are treated as upper bounds, and then `rna()` function performs the *RNA* algorithm. This argument is ignored when `bounds` is set to `NULL`.

Note

this coded was not extensively tested.

Examples

```
A <- c(3000, 4000, 5000, 2000)
M <- c(100, 90, 70, 80) # upper bounds.
rna_rec(total_cost = 190, A = A, bounds = M)
rna_rec(total_cost = 312, A = A, bounds = M)
rna_rec(total_cost = 339, A = A, bounds = M)
rna_rec(total_cost = 340, A = A, bounds = M)
```

---

round_oric *Optimal Rounding under Integer Constraints*

Description

[Experimental]

Usage

```
round_oric(x)
```

Arguments

x	(numeric) a numeric vector.

Value

An integer vector.

References

Cont, R., Heidari, M. (2014). Optimal rounding under integer constraints. doi:10.48550/arXiv.1501.00014

Examples

```
x <- c(4.5, 4.1, 4.9)
round_oric(x) # 4 4 5
```

`SimpleGreedy` *Integer-valued Optimal Univariate Allocation Under Constraints for Stratified Sampling*

Description

[Experimental]

Simple algorithm from paper Friedrich et al. (2015) for integer-valued optimal allocation in stratified sampling.

Usage

```
SimpleGreedy(
  n,
  Ah,
  mh = rep(1, length(Ah)),
  Mh = rep(Inf, length(Ah)),
  nh = mh
)

SimpleGreedy2(v0, Nh, Sh, mh = rep(1, length(Nh)), Mh = Nh, nh = mh)
```

Arguments

n	• target sample size for allocation.
Ah	• population strata sizes * standard deviations of a given variable in strata.
mh	• lower constraints for sample sizes in strata.
Mh	• upper constraints for sample sizes in strata.
nh	• initial allocation (if not given then nh=mh).
v0	• upper limit for value of variance which must be attained for computed optimal allocation.
Nh	• population strata sizes.
Sh	• standard deviations of a given variable in strata.

Value

A vector of optimal allocation sizes.

Functions

- `SimpleGreedy2()`:

References

Friedrich, U., Münnich, R., de Vries, S. and Wagner, M. (2015) Fast integer-valued algorithms for optimal allocations under constraints in stratified sampling, *Computational Statistics and Data Analysis*, 92, pp. 1–12. https://www.sciencedirect.com/science/article/pii/S0167947315001413

---

var_st *Variance of the Stratified Estimator*

Description

[Stable]

Compute the value of the variance function V of the stratified estimator, which is of the following generic form:

$$\sum_{h=1}^{H} \frac{A_h^2}{x_h} - A_0,$$

where H denotes total number of strata, $x_1, \ldots, x_H$ are strata sample sizes and A_0, $A_h > 0$, $h = 1, \ldots, H$, are population constants.

Usage

```
var_st(x, A, A0)

var_st_tsi(x, N, S)
```

Arguments

`x`	(`numeric`) sample allocations $x_1, \ldots, x_H$ in strata.
`A`	(`numeric`) population constants $A_1, \ldots, A_H$.
`A0`	(`number`) population constant A_0.
`N`	(`numeric`) strata sizes $N_1, \ldots, N_H$.
`S`	(`numeric`) strata standard deviations of a given study variable $S_1, \ldots, S_H$.

Value

Value of the variance V for a given allocation vector $x_1, \ldots, x_H$.

Functions

- `var_st_tsi()`: computes value of variance V for the case of *stratified* π *estimator* of the population total and *stratified simple random sampling without replacement* design. This particular case yields:

$$A_h = N_h S_h, \quad h = 1, \ldots, H,$$

$$A_0 = \sum_{h=1}^{H} N_h S_h^2,$$

where N_h is the size of stratum h, and S_h is stratum standard deviation of a study variable, $h = 1, \ldots, H$.

References

Särndal, C.-E., Swensson, B. and Wretman, J. (1992). *Model Assisted Survey Sampling*, Chapter 3.7 *Stratified Sampling*, Springer, New York.

Examples

```
N <- c(3000, 4000, 5000, 2000)
S <- rep(1, 4)
M <- c(100, 90, 70, 80)
xopt <- opt(n = 190, A = N * S, M = M)
var_st_tsi(x = xopt, N, S) # 1017579
```

Index