Package 'MOSAlloc'

June 7, 2026

Title Constraint Multiobjective Sample Allocation

Version 1.2.6

Description Provides a framework for multipurpose optimal resource allocation in survey sampling, extending the classical optimal allocation principles introduced by Tschuprow (1923) and Neyman (1934) to multidomain and multivariate allocation problems. The primary method mosalloc() allows for the consideration of precision and cost constraints at the subpopulation level while minimizing either a vector of sampling errors or survey costs across a broad range of optimal sample allocation problems. The approach supports both single- and multistage designs. For single-stage stratified random sampling, the mosallocSTRS() function offers a user-friendly interface. Sensitivity analysis is supported through the problem's dual variables, which are naturally obtained via the internal use of the Embedded Conic Solver from the 'ECOSolveR' package. See Willems (2025, <doi:10.25353/ubtr-9200-484c-5c89>) for a detailed description of the theory behind 'MOSAlloc'.

License GPL (>= 3)

URL https://gitlab.com/willemsf/mosalloc

BugReports https://gitlab.com/willemsf/mosalloc/-/issues

Encoding UTF-8

RoxygenNote 7.3.3

Imports ECOSolveR, Matrix

Suggests parallel, testthat (>= 3.0.0)

NeedsCompilation no

Maintainer Felix Willems <mail.willemsf+MOSAlloc@gmail.com>

Config/testthat/edition 3

Author Felix Willems [aut, cre] (ORCID:
<https://orcid.org/0009-0005-2987-5825>),
Ralf Münnich [ths] (ORCID: <https://orcid.org/0000-0001-8285-5667>)

Repository CRAN

Date/Publication 2026-06-07 06:00:02 UTC

Contents

`constructArestrSTRS` *Constructor for precision constraints*

Description

A helper function for generating precision matrix `A` and corresponding right-hand side `a` under stratified random sampling (STRS) as input to the multiobjective allocation function `mosalloc()`.

Usage

```
constructArestrSTRS(X_var, X_tot, N, list, fpc = TRUE)
```

Arguments

`X_var`	(type: `matrix`) A matrix containing stratum- (rows) and variable- (columns) specific precision units.
`X_tot`	(type: `matrix`) A matrix containing stratum- (rows) and variable- (columns) specific totals.
`N`	(type: `vector`) A vector of stratum sizes.
`list`	(type: `list`) A list of lists taking subpopulation- (domain/area) specific arguments. Elements are lists containing the following components which in turn correspond to one specific precision restriction: `..$stratum_id` (type: `numeric`) A vector containing the indices of the strata considered for the current restriction. The indices must coincide with the corresponding row numbers of `X_var` and `X_tot`. `..$variate` (type: `character` or `numeric`) The column name or column index of `X_var` to be addressed. `..$measure` (type: `character`) "RSE" (relative standard error) or "VAR" (variance). `..$bound` (type: `numeric`) An upper bound to "RSE" (or "VAR"). `..$name` (type: `character`) The name of the subpopulation (domain/area).
`fpc`	(type: `logical`) A `TRUE` or `FALSE` statement indicating whether the finite population correction should be considered.

Value

The function `constructArestrSTRS()` returns a list containing

`A` (type: `matrix`) a precision matrix for the quality restrictions and

`a` (type: `vector`) a precision vector for the corresponding right-hand side usable as input to the multiobjective allocation function `mosalloc()`.

Examples

```
# Artificial population of 50 568 business establishments and 5 business
# sectors (data from Valliant, R., Dever, J. A., & Kreuter, F. (2013).
# Practical tools for designing and weighting survey samples. Springer.
# https://doi.org/10.1007/978-1-4614-6449-5, Example 5.2 pages 133-9)

# See also https://umd.app.box.com/s/9yvvibu4nz4q6rlw98ac/file/297813512360
# file: Code 5.3 constrOptim.example.R

Nh <- c(6221, 11738, 4333, 22809, 5467) # stratum sizes

# Revenues
mh.rev <- c(85, 11, 23, 17, 126) # mean revenue
Sh.rev <- c(170.0, 8.8, 23.0, 25.5, 315.0) # standard deviation revenue

# Employees
mh.emp <- c(511, 21, 70, 32, 157) # mean number of employees
Sh.emp <- c(255.50, 5.25, 35.00, 32.00, 471.00) # std. dev. employees

# Proportion of establishments claiming research credit
ph.rsch <- c(0.8, 0.2, 0.5, 0.3, 0.9)

# Proportion of establishments with offshore affiliates
ph.offsh <- c(0.06, 0.03, 0.03, 0.21, 0.77)

# Matrix containing stratum-specific variance components
X_var <- cbind(Sh.rev**2,
               Sh.emp**2,
               ph.rsch * (1 - ph.rsch) * Nh/(Nh - 1),
               ph.offsh * (1 - ph.offsh) * Nh/(Nh - 1))
colnames(X_var) <- c("rev", "emp", "rsch", "offsh")

# Matrix containing stratum-specific totals
X_tot <- cbind(mh.rev, mh.emp, ph.rsch, ph.offsh) * Nh
colnames(X_tot) <- c("rev", "emp", "rsch", "offsh")

# Examples
#-------------------------------------------------------------------------
# Example 1: Assume we require at maximum 5 % relative standard error (RSE)
# for estimates of the proportion of businesses with offshore affiliates.
#
# The input \code{A} and \code{a} to \code{mosalloc} can be calculated as
# follows:
A <- matrix(ph.offsh * (1 - ph.offsh) * Nh**3/(Nh - 1)/sum(Nh * ph.offsh)**2,
```

```
nrow = 1)
a <- sum(ph.offsh * (1 - ph.offsh) * Nh**2/(Nh - 1)
)/sum(Nh * ph.offsh)**2 + 0.05**2

# Using \code{constructArestrSTRS()} this can also be done via
list <- list(list(stratum_id = 1:5, variate = "offsh", measure = "RSE",
                  bound = 0.05, name = "pop"))
Ac <- constructArestrSTRS(X_var, X_tot, Nh, list, fpc = TRUE)

# or equivalently by
list <- list(list(stratum_id = 1:5, variate = 4, measure = "RSE",
                  bound = 0.05, name = "pop"))
Ac <- constructArestrSTRS(X_var, X_tot, Nh, list, fpc = TRUE)

# Evaluation of the output
Ac$A - A
Ac$a - a

# Example 2: Assume we require at maximum 5 % relative standard error for
# estimates of the proportion of businesses with offshore affiliates and
# for estimates of the proportion of businesses claiming research credit
# separately for strata 1:2 and 3:5 each.

list <- list(list(stratum_id = 1:2, variate = "offsh", measure = "RSE",
                  bound = 0.05, name = "D1"),
             list(stratum_id = 3:5, variate = "offsh", measure = "RSE",
                  bound = 0.05, name = "D2"),
             list(stratum_id = 1:2, variate = "rsch", measure = "RSE",
                  bound = 0.05, name = "D1"),
             list(stratum_id = 3:5, variate = "rsch", measure = "RSE",
                  bound = 0.05, name = "D2"))
Ac <- constructArestrSTRS(X_var, X_tot, Nh, list, fpc = TRUE)
```

`constructCrestrSTRS` *Constructor for cost constraints*

Description

A helper function for generating cost coefficient matrix `C` and corresponding right-hand side `c` under stratified random sampling (STRS) as input to the the multiobjective allocation function `mosalloc()`.

Usage

```
constructCrestrSTRS(H, list)
```

Arguments

H	(type: numeric) The number of strata.
list	(type: list) A list of lists taking subpopulation- (domain/area) specific arguments. Elements are lists containing the following components which correspond to one specific cost restriction: ..$stratum_id (type: numeric) A vector containing the indices of the strata considered for the current restriction. The indices must coincide with the row numbers of X_var and X_tot. ..$c_coef (type: numeric) A vector of length length(stratum_id) containing the stratum-specific cost components for the set of strata that is going to be bounded by above and/or below. ..$c_upper (type: numeric) The cost upper bound value. NULL if not present. ..$c_lower (type: numeric) The cost lower bound value. NULL if not present. ..$name (type: character) The name of the subpopulation (domain/area).

Value

The function constructCrestrSTRS() returns a list

C (type: matrix): a cost matrix for the cost restrictions and

c (type: vector): a cost vector for the corresponding right-hand side
usable as input to the multiobjective allocation function mosalloc().

Examples

```
# Artificial population of 50 568 business establishments and 5 business
# sectors (data from Valliant, R., Dever, J. A., & Kreuter, F. (2013).
# Practical tools for designing and weighting survey samples. Springer.
# https://doi.org/10.1007/978-1-4614-6449-5, Example 5.2 pages 133-9)

# See also https://umd.app.box.com/s/9yvvibu4nz4q6rlw98ac/file/297813512360
# file: Code 5.3 constrOptim.example.R

Nh <- c(6221, 11738, 4333, 22809, 5467) # stratum sizes
H <- length(Nh)
ch <- c(120, 80, 80, 90, 150) # stratum-specific cost of surveying
budget <- 300000

# Examples
#-------------------------------------------------------------------------
# Example 1: Assume we want so specify one overall cost constraint for the
# five strata. The cost of surveying must not exceed 300000 $.

# The input \code{C} and \code{C} to \code{mosalloc} can be specified as
# follows:

C <- matrix(ch, nrow = 1)
c <- as.vector(budget)

# Using \code{constructCrestrSTRS} this can also be done via
list <- list(list(stratum_id = 1:5, c_coef = ch, c_lower = NULL,
```

```
                  c_upper = budget, name = "Overall"))
Cc <- constructCrestrSTRS(H, list)

# Evaluation of the output
Cc$C - C
Cc$c - c

# Example 2: In addition to the overall cost constraint from Example 1,
# we want to specify a minimum sample size for strata 1 to 3.

# The input \code{C} and \code{C} to \code{mosalloc} can be specified as
# follows:

C <- rbind(ch,
           ch * c(-1, -1, -1, 0, 0))
c <- c(budget,                          # Maximum overall survey budget
       - 0.5 * budget)                  # Minimum overall budget for strata 1-3

# Using \code{constructCrestrSTRS} this can also be done via
list <- list(list(stratum_id = 1:5, c_coef = ch, c_lower = NULL,
                  c_upper = budget, name = "Overall"),
             list(stratum_id = 1:3, c_coef = ch[1:3], c_lower = 0.5 * budget,
                  c_upper = NULL, name = "1to3"))
Cc <- constructCrestrSTRS(H, list)

# Evaluation of the output
Cc$C - C
Cc$c - c
```

`constructDobjCostSTRS` *Constructor for cost objective components*

Description

A helper function for generating cost matrix `D` and fixed cost vector `d` under stratified random sampling (STRS) as input to the multiobjective allocation function `mosalloc()`.

Usage

```
constructDobjCostSTRS(X_cost, X_fixed, list)
```

Arguments

`X_cost`	(type: `matrix`) A matrix containing stratum- (rows) and type- (columns) specific cost coefficients associated with fixed cost. Types of cost might be, e.g. '$ US', 'minutes', 'sample size', etc.
`X_fixed`	(type: `matrix`) A matrix containing stratum- (rows) and type- (columns) specific cost coefficients associated with fixed cost.

`list` (type: `list`) A list of lists taking subpopulation- (domain/area) specific arguments. Elements are lists containing the following components corresponding to one specific cost type:
`..$stratum_id` (type: `numeric`) A vector containing the indices of the strata considered for the current objective. The indices must coincide with the row numbers of `X_cost`.
`..$c_type` (type: `character` or `numeric`) The column name or column index of `X_cost` to be addressed.
`..$name` (type: `character`) The name of the subpopulation (domain/area).

Value

The function `constructDobjCostSTRS()` returns a list containing

`$D` (type: `matrix`): the cost coefficient matrix for cost objectives and

`$d` (type: `vector`): the vector of fixed costs
usable as input to the multiobjective allocation function `mosalloc()`.

Examples

```
# Assume we are given two regions stratified into three strata each. We now
# might balance the cost of surveying between both regions.

# Stratum-specific variable cost
ch <- c(25, 40, 33, 18, 53, 21)
names(ch) <- c("R1_S1", "R1_S2", "R1_S3",
               "R2_S1", "R2_S2", "R2_S3")

# Stratum-specific fixed cost
cf <- c(55, 50, 55, 50, 55, 50)
names(cf) <- c("R1_S1", "R1_S2", "R1_S3",
               "R2_S1", "R2_S2", "R2_S3")

# The input \code{D} and \code{d} to \code{mosalloc()} can be specified as
# follows:

D <- matrix(c(ch[1:3], rep(0, 6), ch[4:6]), 2, 6, byrow = TRUE)
d <- as.vector(c(sum(cf[1:3]), sum(cf[4:6])))

# Using \code{constructDobjCostSTRS()} this can also be done via

X_cost <- matrix(ch, ncol = 1)
colnames(X_cost) <- "$ US"

X_fixed <- matrix(cf, ncol = 1)
colnames(X_fixed) <- "$ US"

list <- list(list(stratum_id = 1:3, c_type = "$ US", name = "R1"),
             list(stratum_id = 4:6, c_type = "$ US", name = "R2"))
Dc <- constructDobjCostSTRS(X_cost, X_fixed, list)

# Evaluation of the output
```

```
Dc$D - D
Dc$d - d
```

`constructDobjPrecisionSTRS` *Constructor for precision objective components*

Description

A helper function for generating precision matrix `D` and finite population correction `d` under stratified random sampling (STRS) as input to the multiobjective allocation function `mosalloc()`.

Usage

```
constructDobjPrecisionSTRS(X_var, X_tot, N, list, fpc = TRUE)
```

Arguments

`X_var`	(type: `matrix`) A matrix containing stratum- (rows) and variable- (columns) specific precision units.
`X_tot`	(type: `matrix`) A matrix containing stratum- (rows) and variable- (columns) specific totals.
`N`	(type: `vector`) A vector of stratum sizes.
`list`	(type: `list`) A list of lists taking subpopulation- (domain/area) specific arguments. Elements are lists containing the following components which in turn correspond to one specific precision restriction: `..$stratum_id` (type: `numeric`) A vector containing the indices of the strata considered for the current objective component. The indices must coincide with the corresponding row numbers of `X_var` and `X_tot`. `..$variate` (type: `character` or `numeric`) The column name or column index of `X_var` to be addressed. `..$measure` (type: `character` or `numeric`) As character "relVAR" (relative variance) or "VAR" (variance). A numerical between 0 and 1 indicates a gradation coefficient that balances between "relVar" (..$measure = 0) and "VAR" (..$measure = 1). `..$name` (type: `character`) The name of the subpopulation (domain/area).
`fpc`	(type: `logical`) A `TRUE` or `FALSE` statement indicating whether the finite population correction should be considered.

Value

The function `constructDobjPrecisionSTRS()` returns a list containing

`$D` (type: `matrix`): the precision matrix for quality objectives and

`$d` (type: `vector`): the vector of finite population corrections
usable as input to the multiobjective allocation function `mosalloc()`.

Examples

```
# Artificial population of 50 568 business establishments and 5 business
# sectors (data from Valliant, R., Dever, J. A., & Kreuter, F. (2013).
# Practical tools for designing and weighting survey samples. Springer.
# https://doi.org/10.1007/978-1-4614-6449-5, Example 5.2 pages 133-9)

# See also https://umd.app.box.com/s/9yvvibu4nz4q6rlw98ac/file/297813512360
# file: Code 5.3 constrOptim.example.R

Nh <- c(6221, 11738, 4333, 22809, 5467) # stratum sizes

# Revenues
mh.rev <- c(85, 11, 23, 17, 126) # mean revenue
Sh.rev <- c(170.0, 8.8, 23.0, 25.5, 315.0) # standard deviation revenue

# Employees
mh.emp <- c(511, 21, 70, 32, 157) # mean number of employees
Sh.emp <- c(255.50, 5.25, 35.00, 32.00, 471.00) # std. dev. employees

# Proportion of estabs claiming research credit
ph.rsch <- c(0.8, 0.2, 0.5, 0.3, 0.9)

# Proportion of estabs with offshore affiliates
ph.offsh <- c(0.06, 0.03, 0.03, 0.21, 0.77)

budget <- 300000 # overall available budget
n.min  <- 100 # minimum stratum-specific sample size

# Matrix containing stratum-specific variance components
X_var <- cbind(Sh.rev**2,
               Sh.emp**2,
               ph.rsch * (1 - ph.rsch) * Nh/(Nh - 1),
               ph.offsh * (1 - ph.offsh) * Nh/(Nh - 1))
colnames(X_var) <- c("rev", "emp", "rsch", "offsh")

# Matrix containing stratum-specific totals
X_tot <- cbind(mh.rev, mh.emp, ph.rsch, ph.offsh) * Nh
colnames(X_tot) <- c("rev", "emp", "rsch", "offsh")

# Examples
#-------------------------------------------------------------------------
# Example 1: Assume we want to minimize the variation of estimates for
# revenue.
#
# The input \code{D} and \code{d} to \code{mosalloc()} can be calculated as
# follows:

D <- matrix(Sh.rev**2 * Nh**2, nrow = 1) # objective variance components
d <- sum(Sh.rev**2 * Nh) # finite population correction

# Using \code{constructDobjPrecisionSTRS()} this can also be done via
list <- list(list(stratum_id = 1:5, variate = "rev", measure = "VAR",
```

```
                  name = "pop"))
Dc <- constructDobjPrecisionSTRS(X_var, X_tot, Nh, list, fpc = TRUE)

# or equivalently by
list <- list(list(stratum_id = 1:5, variate = 1, measure = "VAR",
                  name = "pop"))
Dc <- constructDobjPrecisionSTRS(X_var, X_tot, Nh, list, fpc = TRUE)

# Evaluate output
Dc$D - D
Dc$d - d

# Example 2: Minimization of the maximum coefficient of variation of
# estimates for the total revenue, the number of employee, the number of
# businesses claimed research credit, and the number of businesses with
# offshore affiliates

# The input \code{D} and \code{d} to \code{mosalloc()} can be calculated as
# follows:

D <- rbind(Sh.rev**2 * Nh**2 / sum(Nh * mh.rev)**2,
           Sh.emp**2 * Nh**2 / sum(Nh * mh.emp)**2,
           ph.rsch * (1 - ph.rsch) * Nh**3/(Nh - 1)/sum(Nh * ph.rsch)**2,
           ph.offsh * (1 - ph.offsh) * Nh**3/(Nh - 1)/sum(Nh * ph.offsh)**2)
d <- as.vector(D %*% (1 / Nh)) # finite population correction

# Using \code{constructDobjPrecisionSTRS()} this can also be done via
list <- list(list(stratum_id = 1:5, variate = "rev", measure = "relVAR",
                  name = "pop"),
             list(stratum_id = 1:5, variate = "emp", measure = "relVAR",
                  name = "pop"),
             list(stratum_id = 1:5, variate = "rsch", measure = "relVAR",
                  name = "pop"),
             list(stratum_id = 1:5, variate = "offsh", measure = "relVAR",
                  name = "pop"))
Dc <- constructDobjPrecisionSTRS(X_var, X_tot, Nh, list, fpc = TRUE)

# Evaluation of the output
Dc$D - D
Dc$d - d
```

`mosalloc` *Multiobjective sample allocation for constraint multivariate and multidomain optimal allocation in survey sampling*

Description

Computes solutions to standard sample allocation problems under various precision and cost restrictions. The input data is transformed and parsed to the Embedded COnic Solver (ECOS) from

the 'ECOSolveR' package. Multiple survey purposes can be optimized simultaneously through a weighted Chebyshev minimization. Note that in the case of multiple objectives, `mosalloc()` does not necessarily lead to Pareto optimality. This highly depends on the problem structure. A strong indicator for Pareto optimality is when the weighted objective values given by `Dbounds` are constant over all objective components or when all components of `Qbounds` equal `1`. In addition, `mosalloc()` can handle twice-differential convex decision functionals (in which case Pareto optimality is ensured). `mosalloc()` returns dual variables, enabling a detailed sensitivity analysis.

Usage

```
mosalloc(
  D,
  d,
  A = NULL,
  a = NULL,
  C = NULL,
  c = NULL,
  l = 2,
  u = NULL,
  opts = list(sense = "max_precision", f = NULL, df = NULL, Hf = NULL, init_w = 1,
    mc_cores = 1L, pm_tol = 1e-05, max_iters = 100L, print_pm = FALSE)
)
```

Arguments

`D`	(type: `matrix`) The objective matrix. A matrix of either precision or cost units.
`d`	(type: `vector`) The objective vector. A vector of either fixed precision components (e.g. finite population corrections) or fixed costs.
`A`	(type: `matrix`) A matrix of precision units for precision constraints.
`a`	(type: `vector`) The right-hand side vector of the precision constraints.
`C`	(type: `matrix`) A matrix of cost coefficients for cost constraints
`c`	(type: `vector`) The right-hand side vector of the cost constraints.
`l`	(type: `vector`) A vector of lower box constraints.
`u`	(type: `vector`) A vector of upper box constraints.
`opts`	(type: `list`) The options used by the algorithms: `$sense` (type: `character`) Sense of optimization (default = `"max_precision"`, alternative `"min_cost"`). `$f` (type: `function`) Decision functional over the objective vector (default = `NULL`). `$df` (type: `function`) The gradient of f (default = NULL). `$Hf` (type: `function`) The Hesse matrix of f (default = NULL). `$init_w` (type: `numeric` or `matrix`) Preference weightings (default = `1`; The weight for first objective component must be 1). `$mc_cores` (type: `integer`) The number of cores for parallelizing multiple input weightings stacked rowwise (default = `1L`). `$pm_tol` (type: `numeric`) The tolerance for the projection method (default =

1e-5).
max_iters (type: integer) The maximum number of iterations (default = 100L).
$print_pm (type: logical) A TRUE or FALSE statement whether iterations of the projection method should be printed (default = FALSE).

Value

mosalloc() returns a list containing the following components:

$w The initial preference weighting opts$init_w.

$n The vector of optimal sample sizes.

$J The optimal objective vector.

$Objective The objective value with respect to decision functional f. NULL if opts$f = NULL.

$Utopian The component-wise univariate optimal objective vector. NULL if opts$f = NULL.

$Normal The vector normal to the Pareto frontier at $J.

$dfJ The gradient of opts$f evaluated at $J.

$Sensitivity The dual variables of the objectives and constraints.

$Qbounds The Quality bounds of the Lorentz cones.

$Dbounds The weighted objective constraints ($w * $J).

$Scalepar An internal scaling parameter.

$Ecosolver A list of ECOSolveR returns including:
...$Ecoinfostring The info string of ECOSolveR::ECOS_csolve().
...$Ecoredcodes The redcodes of ECOSolveR::ECOS_csolve().
...$Ecosummary Problem summary of ECOSolveR::ECOS_csolve().

$Timing Run time info.

$Iteration A list of internal iterates. NULL if opts$f = NULL.

Note

Precision optimization *(*opts$sense == "max_precision", opts$f == NULL*)*

The mathematical problem solved is

$$\min_{n,z,t}\{t : Dz - d \leq w^{-1}t, Az \leq a, Cn \leq c, 1 \leq n_i z_i \, \forall i, l \leq n \leq u\}$$

with $\mathbf{1} \leq l \leq u \leq N$, where

- $n, z \in \mathbb{R}^m$ and $t \in \mathbb{R}$ are the optimization variables,
- $D \in \{M \in \mathbb{R}^{k_D \times m} : \sum_i M_{ij} > 0 \, \forall j, \sum_j M_{ij} > 0 \, \forall i\}$ a matrix of nonnegative objective precision components with k_D the number of precision objectives (the number of variables of interest),
- $d \in \mathbb{R}^{k_D}$ the objective right hand-side (RHS), e.g. the finite population correction (fpc),
- $A \in \mathbb{R}^{k_A \times m}$ a nonnegative precision matrix with k_A the number of precision constraints, $a \in \mathbb{R}^{k_A}$ the corresponding RHS, e.g. fpc + (upper bound to the coefficient of variation)^2,
- $C \in \mathbb{R}^{k_C \times m}$ a cost matrix with k_C the number of cost constraint,

- $c \in \mathbb{R}^{k_C}$ the corresponding RHS,
- $l, u \in \mathbb{R}^m$ the bounds to the sample size vector n,
- $N \in \mathbb{N}^m$ the vector of population sizes, and
- w a given strictly positive preference weighting.

Special cases of this formulation are

- Neyman-Tschuprow allocation (Neyman, 1934 and Tschuprow, 1923):

$$\min_n \Big\{ \sum_{h=1}^{H} \Big(\frac{N_h^2 S_h^2}{n_h} - N_h S_h^2 \Big) : \sum_{h=1}^{H} n_h \leq c \Big\} \quad \Leftrightarrow \quad \min_{n,z,t} \{ t : Dz - d \leq t, Cn \leq c, 1 \leq n_i z_i \, \forall i \}$$

  with $D = (N_1^2 S_1^2, \ldots, N_H^2 S_H^2)$, $d = \sum_{h=1}^{H} N_h S_h^2$, $C = (1, \ldots, 1)$ and c a maximum sample size. Here, H is the number of strata, N_h the size of stratum h and S_h^2 the variance of the variable of interest in stratum h.

- box-constrained optimal allocation:

$$\min_{n,z,t} \{ t : Dz - d \leq t, Cn \leq c, 1 \leq n_i z_i \, \forall i, l \leq n \leq u \}$$

  with $D = (N_1^2 S_1^2, \ldots, N_H^2 S_H^2)$, $d = \sum_{h=1}^{H} N_h S_h^2$, $C = (1, \ldots, 1)$ and c a maximum sample size (cf. Srikantan, 1963 and Münnich et al., 2012). Here, l and u are bounds to the optimal sample size vector.

- cost and precision constrained univariate optimal allocation:

$$\min_{n,z,t} \{ t : Dz - d \leq t, Az \leq a, Cn \leq c, 1 \leq n_i z_i \, \forall i, l \leq n \leq u \}$$

  with $k_D = 1$ (cf. Willems, 2025, Chapter 3).

- multivariate optimal allocation with weighted sum scalarization:

$$\min_{n,z,t} \{ t : w^\top Dz - w^\top d \leq t, Az \leq a, Cn \leq c, 1 \leq n_i z_i \, \forall i, l \leq n \leq u \}$$

  where $w \in \mathbb{R}^{k_D}$ is a strictly positive preference weighting (cf. Folks and Antle, 1965, and Rupp, 2018). Note that for this case the problem reduces to cost and precision constrained univariate optimal allocation. Solutions are ensured to be optimal in the Pareto sense.

- box-constraint two-stage cluster sampling:

$$\min_{n_{\mathbf{I}}, n_{\mathbf{II}}} \Big\{ \Big(\frac{N_{\mathbf{I}}^2 S_{\mathbf{I}}^2}{n_{\mathbf{I}}} - N_{\mathbf{I}} S_{\mathbf{I}}^2 \Big) + \frac{N_{\mathbf{I}}}{n_{\mathbf{I}}} \sum_{j=1}^{N_{\mathbf{I}}} \Big(\frac{N_{\mathbf{II}j}^2 S_{\mathbf{II}j}^2}{n_{\mathbf{II}j}} - N_{\mathbf{II}j} S_{\mathbf{II}j}^2 \Big) : c_{\mathbf{I}} n_{\mathbf{I}} + \frac{n_{\mathbf{I}}}{N_{\mathbf{I}}} \sum_{j=1}^{N_{\mathbf{I}}} c_{\mathbf{II}j} n_{\mathbf{II}j} \leq c_{\max},$$

$$l_{\mathbf{I}} \leq n_{\mathbf{I}} \leq u_{\mathbf{I}}, l_{\mathbf{II}} \leq n_{\mathbf{II}} \leq u_{\mathbf{II}} \Big\}$$

  $$\Leftrightarrow \quad \min_{n,z,t} \{ t : Dz - d \leq t, Cn \leq c, 1 \leq n_i z_i \, \forall i, l \leq n \leq u \}$$

  with $D = (N_{\mathbf{I}}^2 S_{\mathbf{I}}^2 - N_{\mathbf{I}} \sum_{j=1}^{N_{\mathbf{I}}} N_{\mathbf{II}j} S_{\mathbf{II}j}^2, N_{\mathbf{I}} N_{\mathbf{II}1}^2 S_{\mathbf{II}1}^2, \ldots, N_{\mathbf{I}} N_{\mathbf{II}N_{\mathbf{I}}}^2 S_{\mathbf{II}N_{\mathbf{I}}}^2)$, $d = N_{\mathbf{I}} S_{\mathbf{I}}^2$, $C = [C_1, C_2]$, where $C_1 = (c_{\mathbf{I}}, l_{\mathbf{II}}^\top, -u_{\mathbf{II}}^\top)^\top$ and $C_2 = [N_{\mathbf{I}}^{-1} c_{\mathbf{II}}^\top; -\mathbf{I}; \mathbf{I}]$ ($\mathbf{I}$ is the identity matrix), $c = (c_{\max}, 0, \ldots, 0)^\top$, where $l = (l_{\mathbf{I}}, l_{\mathbf{I}} l_{\mathbf{II}}^\top)^\top$ and $u = (u_{\mathbf{I}}, u_{\mathbf{I}} u_{\mathbf{II}}^\top)^\top$ (cf. Willems, 2025,

Chapter 3). Here, $N_{\mathbf{I}}$ is the number of clusters, $N_{\mathbf{II}j}$ the size of cluster j, $S^2_{\mathbf{I}}$ the between cluster variance and $S^2_{\mathbf{II}j}$ the within cluster variances of the variable of interest. Furthermore, c_{max} is a maximum expected cost, $c_{\mathbf{I}}$ a variable cost for sampling one cluster, and $c_{\mathbf{II}j}$ a variable cost for sampling one unit in cluster j. The optimal number of clusters to be drawn and the optimal sample sizes are given through $n = (n_{\mathbf{I}}, n_{\mathbf{I}}n_{\mathbf{II}1}, \ldots, n_{\mathbf{I}}n_{\mathbf{II}N_{\mathbf{I}}})^\top$.

For the special cases above, solutions are unique and, thus, Pareto optimal. For the general multi-objective problem formulation this is not the case. However, a strong indicator for uniqueness of solutions is $n_i z_i = 1 \, \forall i$ (`Qbounds`) or $Dz - d = w^{-1}t$ (`Dbounds`). Uniqueness can be ensured via a stepwise procedure implemented in `mosallocStepwiseFirst()`.

Precision optimization with nonlinear decision functional *(*`opts$sense == "max_precision"`*,* `opts$f ==`f*,* `opts$f == `∇f*,* `opts$f == `Hf*)*

The mathematical problem solved is

$$\min_{n,z}\{f(Dz - d) : Az \leq a, Cn \leq c, 1 \leq n_i z_i \, \forall i, l \leq n \leq u\}$$

with components as specified above and where $f : \mathbb{R}^{k_D} \to \mathbb{R}, x \mapsto f(x)$ is a twice-differentiable convex decision functional. E.g. a p-norm $f(x) = \|x\|_p$ with $p \in \mathbb{N}$.

Cost optimization *(*`opts$sense == "min_cost"`*)*

The mathematical problem solved is

$$\min_{n,z,t}\{t : Dn - d \leq \mathbf{1}t, Az \leq a, Cn \leq c, 1 \leq n_i z_i \, \forall i, l \leq n \leq u\}$$

with $1 \leq l \leq u \leq N$. Hence, the only difference to precision optimization is the type of objective constraint $Dn - d \leq \mathbf{1}t$.

Special cases of this formulation are

- the cost optimal allocation (possibly multivariate, i.e. $k_A \geq 2$):

$$\min_{n,z,t}\{t : Dn - d \leq \mathbf{1}t, Az \leq a, 1 \leq n_i z_i \, \forall i\}$$

  where $D^\top$ is a vector of stratum-specific sampling cost and d some fixed cost.

Numerical caution: extreme heterogeneity

In optimal allocation problems, the components of D determine how much each stratum contributes to the objective function. When the entries of D vary by several orders of magnitude, the problem may become numerically ill-conditioned. In such cases, a small number of strata can dominate the objective, while other strata have only a minor influence on the solution.

Under an inequality constraint such as $\sum_h n_h \leq n_{\max}$, the optimizer may return solutions where the constraint is not tight due to solver tolerances. This can occur when additional sample size do not lead to a numerically meaningful improvement in the objective, given the solver tolerances.

If an exact overall sample size is required, it may be preferable to formulate the sample-size restriction as an equality constraint represented by two inequalities, i.e.

$$C = \begin{pmatrix} 1 & \cdots & 1 \\ -1 & \cdots & -1 \end{pmatrix}, \qquad c = \begin{pmatrix} n_{\max} \\ -n_{\max} \end{pmatrix}.$$

This formulation removes the possibility of slack in the total sample size constraint. In problems with extreme heterogeneity, this can help avoid degenerate allocations where only a subset of strata effectively drives the solution. See Example 10 for an illustration.

References

See:

Folks, J.L., Antle, C.E. (1965). *Optimum Allocation of Sampling Units to Strata when there are R Responses of Interest*. Journal of the American Statistical Association, 60(309), 225-233. doi:10.1080/01621459.1965.10480786.

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

Neyman, J. (1934). *On the Two Different Aspects of the Representative Method: The Method of Stratified Sampling and the Method of Purposive Selection*. Journal of the Royal Statistical Society, 97(4), 558–625.

Tschuprow, A.A. (1923). *On the Mathematical Expectation of the Moments of Frequency Distribution in the Case of Correlated Observations*. Metron, 2(3,4), 461-493, 646-683.

Rupp, M. (2018). *Optimization for Multivariate and Multi-domain Methods in Survey Statistics* (Doctoral dissertation). Trier University. doi:10.25353/UBTR8351543214XX.

Srikantan, K.S. (1963). *A Problem in Optimum Allocation*. Operations Research, 11(2), 265-274.

Willems, F. (2025). *A Framework for Multiobjective and Uncertain Resource Allocation Problems in Survey Sampling based on Conic Optimization* (Doctoral dissertation). Trier University. doi:10.25353/ubtr9200484c5c89.

Examples

```
# Artificial population of 50 568 business establishments and 5 business
# sectors (data from Valliant, R., Dever, J. A., & Kreuter, F. (2013).
# Practical tools for designing and weighting survey samples. Springer.
# https://doi.org/10.1007/978-1-4614-6449-5, Example 5.2 pages 133-9)

# See also <https://umd.app.box.com/s/9yvvibu4nz4q6rlw98ac/file/297813512360>
# file: Code 5.3 constrOptim.example.R

Nh <- c(6221, 11738, 4333, 22809, 5467) # stratum sizes
ch <- c(120, 80, 80, 90, 150) # stratum-specific cost of surveying

# Revenues
mh.rev <- c(85, 11, 23, 17, 126) # mean revenue
Sh.rev <- c(170.0, 8.8, 23.0, 25.5, 315.0) # standard deviation revenue

# Employees
mh.emp <- c(511, 21, 70, 32, 157) # mean number of employees
Sh.emp <- c(255.50, 5.25, 35.00, 32.00, 471.00) # std. dev. employees

# Proportion of estabs claiming research credit
ph.rsch <- c(0.8, 0.2, 0.5, 0.3, 0.9)
```

```
# Proportion of estabs with offshore affiliates
ph.offsh <- c(0.06, 0.03, 0.03, 0.21, 0.77)

budget <- 300000 # overall available budget
n.min  <- 100 # minimum stratum-specific sample size

# Examples
#-------------------------------------------------------------------------
# Example 1: Minimization of the variation of estimates for revenue subject
# to cost restrictions and precision restrictions to the coefficient of
# variation of estimates for the proportion of businesses with offshore
# affiliates.

l <- rep(n.min, 5) # minimum sample size per stratum
u <- Nh            # maximum sample size per stratum
C <- rbind(ch,
           ch * c(-1, -1, -1, 0, 0))
c <- c(budget,                        # Maximum overall survey budget
       - 0.5 * budget)                # Minimum overall budget for strata 1-3

# We require at maximum 5 % relative standard error for estimates of
# proportion of businesses with offshore affiliates
A <- matrix(ph.offsh * (1 - ph.offsh) * Nh**3/(Nh - 1)/sum(Nh * ph.offsh)**2,
 nrow = 1)
a <- sum(ph.offsh * (1 - ph.offsh) * Nh**2/(Nh - 1)
)/sum(Nh * ph.offsh)**2 + 0.05**2

D <- matrix(Sh.rev**2 * Nh**2, nrow = 1) # objective variance components
d <- sum(Sh.rev**2 * Nh) # finite population correction

opts <- list(sense = "max_precision",
             f = NULL, df = NULL, Hf = NULL,
             init_w = 1,
             mc_cores = 1L, pm_tol = 1e-05,
             max_iters = 100L, print_pm = FALSE)

sol <- mosalloc(D = D, d = d, A = A, a = a, C = C, c = c, l = l, u = u,
                opts = opts)

# Check solution statement of the internal solver to verify feasibility
sol$Ecosolver$Ecoinfostring # [1] "Optimal solution found"

# Check constraints
c(C[1, ] %*% sol$n) # [1] 3e+05
c(C[2, ] %*% sol$n) # [1] -150000
c(sqrt(A %*% (1 / sol$n) - A %*% (1 / Nh))) # 5 % rel. std. err.

#-------------------------------------------------------------------------
# Example 2: Minimization of the maximum relative variation of estimates for
# the total revenue, the number of employee, the number of businesses claimed
# research credit, and the number of businesses with offshore affiliates
# subject to cost restrictions
```

```
l <- rep(n.min, 5) # minimum sample size per stratum
u <- Nh            # maximum sample size per stratum
C <- rbind(ch, ch * c(-1, -1, -1, 0, 0))
c <- c(budget, - 0.5 * budget)
A <- NULL # no precision constraint
a <- NULL # no precision constraint

# Precision components (Variance / Totals^2) for multidimensional objective
D <- rbind(Sh.rev**2 * Nh**2/sum(Nh * mh.rev)**2,
           Sh.emp**2 * Nh**2/sum(Nh * mh.emp)**2,
           ph.rsch * (1 - ph.rsch) * Nh**3/(Nh - 1)/sum(Nh * ph.rsch)**2,
           ph.offsh * (1 - ph.offsh) * Nh**3/(Nh - 1)/sum(Nh * ph.offsh)**2)

d <- as.vector(D %*% (1 / Nh)) # finite population correction

opts <- list(sense = "max_precision",
             f = NULL, df = NULL, Hf = NULL,
             init_w = 1,
             mc_cores = 1L, pm_tol = 1e-05,
             max_iters = 100L, print_pm = FALSE)

sol <- mosalloc(D = D, d = d, C = C, c = c, l = l, u = u, opts = opts)

# Obtain optimal objective value
sol$J # [1] 0.0017058896 0.0004396972 0.0006428474 0.0017058896

# Obtain corresponding normal vector
sol$Normal # [1] 6.983112e-01 1.293135e-11 1.543172e-11 3.016888e-01

# => Revenue and offshore affiliates are dominating the solution with a
#    ratio of approximately 2:1 (sol$Normal[1] / sol$Normal[4])

#------------------------------------------------------------------------------
# Example 3: Example 2 with preference weighting

w <- c(1, 3.85, 3.8, 1.3) # preference weighting
l <- rep(n.min, 5) # minimum sample size per stratum
u <- Nh            # maximum sample size per stratum
C <- rbind(ch, ch * c(-1, -1, -1, 0, 0))
c <- c(budget, - 0.5 * budget)
A <- NULL # no precision constraint
a <- NULL # no precision constraint

D <- rbind(Sh.rev**2 * Nh**2/sum(Nh * mh.rev)**2,
           Sh.emp**2 * Nh**2/sum(Nh * mh.emp)**2,
           ph.rsch * (1 - ph.rsch) * Nh**3/(Nh - 1)/sum(Nh * ph.rsch)**2,
           ph.offsh * (1 - ph.offsh) * Nh**3/(Nh - 1)/sum(Nh * ph.offsh)**2)

d <- as.vector(D %*% (1 / Nh))

opts <- list(sense = "max_precision",
             f = NULL, df = NULL, Hf = NULL,
```

```
            init_w = w,
            mc_cores = 1L, pm_tol = 1e-05,
            max_iters = 100L, print_pm = FALSE)

mosalloc(D = D, d = d, C = C, c = c, l = l, u = u, opts = opts)

#-------------------------------------------------------------------------
# Example 4: Example 2 with multiple preference weightings for simultaneous
# evaluation

w <- matrix(c(1.0, 1.0, 1.0, 1.0,       # matrix of preference weightings
              1.0, 3.9, 3.9, 1.3,
              0.8, 4.2, 4.8, 1.5,
              1.2, 3.5, 4.8, 2.0,
              2.0, 1.0, 1.0, 2.0), 5, 4, byrow = TRUE)
w <- w / w[,1]     # rescale w (ensure the first weighting to be one)
l <- rep(n.min, 5) # minimum sample size per stratum
u <- Nh            # maximum sample size per stratum
C <- rbind(ch, ch * c(-1, -1, -1, 0, 0))
c <- c(budget, - 0.5 * budget)
A <- NULL # no precision constraint
a <- NULL # no precision constraint

D <- rbind(Sh.rev**2 * Nh**2/sum(Nh * mh.rev)**2,
           Sh.emp**2 * Nh**2/sum(Nh * mh.emp)**2,
           ph.rsch * (1 - ph.rsch) * Nh**3/(Nh - 1)/sum(Nh * ph.rsch)**2,
           ph.offsh * (1 - ph.offsh) * Nh**3/(Nh - 1)/sum(Nh * ph.offsh)**2)

d <- as.vector(D %*% (1 / Nh))

opts <- list(sense = "max_precision",
             f = NULL, df = NULL, Hf = NULL,
             init_w = w,
             mc_cores = 1L, pm_tol = 1e-05,
             max_iters = 100L, print_pm = FALSE)

sols <- mosalloc(D = D, d = d, C = C, c = c, l = l, u = u, opts = opts)
lapply(sols, function(sol){sol$Qbounds})

#-------------------------------------------------------------------------
# Example 5: Example 2 where a weighted sum scalarization of the objective
# components is minimized

l <- rep(n.min, 5) # minimum sample size per stratum
u <- Nh            # maximum sample size per stratum
C <- matrix(ch, nrow = 1)
c <- budget
A <- NULL # no precision constraint
a <- NULL # no precision constraint

# Objective variance components
D <- rbind(Sh.rev**2 * Nh**2/sum(Nh * mh.rev)**2,
           Sh.emp**2 * Nh**2/sum(Nh * mh.emp)**2,
```

```
            ph.rsch * (1 - ph.rsch) * Nh**3/(Nh - 1)/sum(Nh * ph.rsch)**2,
            ph.offsh * (1 - ph.offsh) * Nh**3/(Nh - 1)/sum(Nh * ph.offsh)**2)

d <- as.vector(D %*% (1 / Nh)) # finite population correction

# Simple weighted sum as decision functional
wss <- c(1, 1, 0.5, 0.5) # preference weighting (weighted sum scalarization)

Dw <- wss %*% D
dw <- as.vector(Dw %*% (1 / Nh))

opts <- list(sense = "max_precision",
             f = NULL, df = NULL, Hf = NULL,
             init_w = 1,
             mc_cores = 1L, pm_tol = 1e-05,
             max_iters = 1000L, print_pm = FALSE)

# Solve weighted sum scalarization (WSS) via mosalloc
sol_wss <- mosalloc(D = Dw, d = dw, C = C, c = c, l = l, u = u, opts = opts)

# Obtain optimal objective values
J <- D %*% (1 / sol_wss$n) - d

# Reconstruct solution via a weighted Chebyshev minimization
wcm <- J[1] / J
opts = list(sense = "max_precision",
            f = NULL, df = NULL, Hf = NULL,
            init_w = matrix(wcm, 1),
            mc_cores = 1L, pm_tol = 1e-05,
            max_iters = 1000L, print_pm = FALSE)

sol_wcm <- mosalloc(D = D, d = d, C = C, c = c, l = l, u = u, opts = opts)

# Compare solutions
rbind(t(J), sol_wcm$J)
#            [,1]         [,2]         [,3]        [,4]
# [1,] 0.00155645 0.0004037429 0.0005934474 0.001327165
# [2,] 0.00155645 0.0004037429 0.0005934474 0.001327165

rbind(sol_wss$n, sol_wcm$n)
#          [,1]     [,2]     [,3]     [,4]     [,5]
# [1,] 582.8247 236.6479 116.7866 839.5988 841.4825
# [2,] 582.8226 236.6475 116.7871 839.5989 841.4841

rbind(wss, sol_wcm$Normal / sol_wcm$Normal[1])
#    [,1]      [,2]      [,3]      [,4]
#wss    1 1.0000000 0.5000000 0.5000000
#       1 0.9976725 0.4997552 0.4997463

#------------------------------------------------------------------------------
# Example 6: Example 1 with two subpopulations and a p-norm as decision
# functional
```

```
l <- rep(n.min, 5) # minimum sample size per stratum
u <- Nh            # maximum sample size per stratum
C <- rbind(ch, ch * c(-1, -1, -1, 0, 0))
c <- c(budget, - 0.5 * budget)

# At maximum 5 % relative standard error for estimates of proportion of
# businesses with offshore affiliates
A <- matrix(ph.offsh * (1 - ph.offsh) * Nh**3/(Nh - 1)/sum(Nh * ph.offsh)**2,
 nrow = 1)
a <- sum(ph.offsh * (1 - ph.offsh) * Nh**2/(Nh - 1)
)/sum(Nh * ph.offsh)**2 + 0.05**2

D <- rbind((Sh.rev**2 * Nh**2)*c(0,0,1,1,0),
           (Sh.rev**2 * Nh**2)*c(1,1,0,0,1))# objective variance components
d <- as.vector(D %*% (1 / Nh)) # finite population correction

# p-norm solution
p <- 5 # p-norm
opts <- list(sense = "max_precision",
             f = function(x) sum(x**p),
             df = function(x) p * x**(p - 1),
             Hf = function(x) diag(p * (p - 1) * x**(p - 2)),
             init_w = 1,
             mc_cores = 1L, pm_tol = 1e-05,
             max_iters = 1000L, print_pm = TRUE)

sol <- mosalloc(D = D, d = d, C = C, c = c, l = l, u = u, opts = opts)

c(sol$Normal/sol$dfJ)/mean(c(sol$Normal/sol$dfJ))
# [1] 0.9999999 1.0000001

#-------------------------------------------------------------------------------
# Example 7: Example 2 with p-norm as decision functional and only one
# overall cost constraint

l <- rep(n.min, 5) # minimum sample size per stratum
u <- Nh            # maximum sample size per stratum
C <- matrix(ch, nrow = 1)
c <- budget
A <- NULL # no precision constraint
a <- NULL # no precision constraint

# Objective precision components
D <- rbind(Sh.rev**2 * Nh**2/sum(Nh * mh.rev)**2,
           Sh.emp**2 * Nh**2/sum(Nh * mh.emp)**2,
           ph.rsch * (1 - ph.rsch) * Nh**3/(Nh - 1)/sum(Nh * ph.rsch)**2,
           ph.offsh * (1 - ph.offsh) * Nh**3/(Nh - 1)/sum(Nh * ph.offsh)**2)

d <- as.vector(D %*% (1 / Nh)) # finite population correction

# p-norm solution
p <- 5 # p-norm
opts <- list(sense = "max_precision",
```

```
              f = function(x) sum(x**p),
              df = function(x) p * x**(p - 1),
              Hf = function(x) diag(p * (p - 1) * x**(p - 2)),
              init_w = 1,
              mc_cores = 1L, pm_tol = 1e-05,
              max_iters = 1000L, print_pm = TRUE)

sol <- mosalloc(D = D, d = d, C = C, c = c, l = l, u = u, opts = opts)

c(sol$Normal/sol$dfJ)/mean(c(sol$Normal/sol$dfJ))
# [1] 1.0014362 0.9780042 1.0197807 1.0007789

#-------------------------------------------------------------------------
# Example 8: Minimization of sample sizes subject to precision constraints

l <- rep(n.min, 5) # minimum sample size per stratum
u <- Nh            # maximum sample size per stratum

# We require at maximum 4.66 % relative standard error for the estimate of
# total revenuee, 5 % for the number of employees, 3 % for the proportion of
# businesses claiming research credit, and 3 % for the proportion of
# businesses with offshore affiliates
A <- rbind(Sh.rev**2 * Nh**2/sum(Nh * mh.rev)**2,
           Sh.emp**2 * Nh**2/sum(Nh * mh.emp)**2,
           ph.rsch * (1 - ph.rsch) * Nh**3/(Nh - 1)/sum(Nh * ph.rsch)**2,
           ph.offsh * (1 - ph.offsh) * Nh**3/(Nh - 1)/sum(Nh * ph.offsh)**2)
a <- as.vector(A%*%(1 / Nh) + c(0.0466, 0.05, 0.03, 0.03)**2)

# We do not consider any additional sample size or cost constraints
C <- NULL # no cost constraint
c <- NULL # no cost constraint

# Since we minimize the sample size, we define D and d as follows:
D <- matrix(1, nrow = 1, ncol = length(Nh)) # objective cost components
d <- as.vector(0)                           # vector of possible fixed cost

opts <- list(sense = "min_cost", # Sense of optimization is survey cost
             f = NULL,
             df = NULL,
             Hf = NULL,
             init_w = 1,
             mc_cores = 1L, pm_tol = 1e-05,
             max_iters = 100L, print_pm = TRUE)

sol <- mosalloc(D = D, d = d, A = A, a = a, l = l, u = u, opts = opts)

sum(sol$n) # [1] 2843.219
sol$J # [1] 2843.219

#-------------------------------------------------------------------------
#-------------------------------------------------------------------------
# Note: Sample size optimization for two-stage cluster sampling can be
#       reduced to the structure of optimal stratified random samplin when
```

```
#       considering expected costs. Therefore, mosalloc() can handle such
#       designs. A benefit is that mosalloc() allows relatively complex
#       sample size restrictions such as box constraints for subsampling.
#       Optimal sample sizes at secondary stages have to be reconstructed
#       from sol$n.
#
# Example 9: Optimal number of primary sampling units (PSU) and secondary
# sampling units (SSU) in 2-stage cluster sampling.

set.seed(1234)
pop <- data.frame(value = rnorm(100, 100, 35),
                  cluster = sample(1:4, 100, replace = TRUE))

CI <- 36  # Sampling cost per PSU
CII <- 10 # Average sampling cost per SSU

NI <- 4                   # Number of PSUs
NII <- table(pop$cluster) # PSU/cluster sizes

S2I <- var(by(pop$value, pop$cluster, sum)) # between PSU variance
S2II <- by(pop$value, pop$cluster, var)     # within PSU variances

D <- matrix(c(NI**2 * S2I - NI * sum(NII * S2II), NI * NII**2 * S2II), 1)
d <- as.vector(NI * S2I) # = D %*% (1 / c(NI, NI * NII))

C <- cbind(c(CI, rep(2, NI), -NII),
           rbind(rep(CII / NI, 4), -diag(4), diag(4)))
c <- as.vector(c(500, rep(0, 8)))

l <- c(2, rep(4, 4))
u <- c(NI, NI * NII)

opts <- list(sense = "max_precision",
             f = NULL,
             df = NULL,
             Hf = NULL,
             init_w = 1,
             mc_cores = 1L, pm_tol = 1e-05,
             max_iters = 100L, print_pm = TRUE)

sol <- mosalloc(D = D, d = d, C = C, c = c, l = l, u = u, opts = opts)

# Optimum number of clusters to be drawn
sol$n[1] # [1] 2.991551

# Optimum number of elements to be drawn within clusters
sol$n[-1] / sol$n[1] # [1] 12.16454 11.60828 15.87949 12.80266

#------------------------------------------------------------------------------
#------------------------------------------------------------------------------
# Example 10 (cf. ''Numerical caution: extreme heterogeneity'' above):
# 15-stratum Neyman-Tschuprow allocation with equal costs
Nh <- c(2,1000,800,1200,950,1100,700,600,500,400,300,200,150,100,50)
```

```
Sh <- c(100,0.0001,0.2,0.4,0.5,0.7,1,1.2,1.5,2,2.5,3,4,5,6)
n_max <- sum(Nh) * 0.9 # Maximum overall sample size

# Neyman-Tschuprow Allocation
nh <- n_max * (Nh * Sh) / sum(Nh * Sh)

# Specify components for mosalloc()
D <- matrix((Nh * Sh)**2, nrow = 1)
d <- as.vector(D %*% (1 / Nh))
A <- NULL
a <- NULL
C1 <- matrix(1, nrow = 1, ncol = ncol(D))
c1 <- n_max
l <- rep(1, length(Nh))
u <- Nh
opts <- list(sense = "max_precision", f = NULL, df = NULL, Hf = NULL,
             init_w = 1, mc_cores = 1L, pm_tol = 1e-05, max_iters = 100L,
             print_pm = FALSE)

# Solve the problem with inequality cost constraint
res1 <- mosalloc(D, d, A, a, C1, c1, l, u, opts)

# Respecify cost components to enforce equality constraints
C2 <- matrix(c(1, -1), nrow = 2, ncol = ncol(D))
c2 <- c(n_max, -n_max)

# Solve the problem with explicit equality cost constraint
res2 <- mosalloc(D, d, A, a, C2, c2, l, u, opts)

# Comparison of results
out <- cbind(nh, res1$n, res2$n, Nh,  (Nh * Sh)**2)
colnames(out) <- c("NT-Allocation", "Inequality", "Equality",
                   "Nh", "(Nh * Sh)^2")
out
#      NT-Allocation Inequality Equality   Nh (Nh * Sh)^2
# [1,]  185.69396933    2.00000      2.0    2    40000.00
# [2,]    0.09284698   56.80793    194.8 1000        0.01
# [3,]  148.55517546  800.00000    800.0  800    25600.00
# [4,]  445.66552639 1200.00000   1200.0 1200   230400.00
# [5,]  441.02317715  950.00000    950.0  950   225625.00
# [6,]  714.92178191 1100.00000   1100.0 1100   592900.00
# [7,]  649.92889265  700.00000    700.0  700   490000.00
# [8,]  668.49828958  600.00000    600.0  600   518400.00
# [9,]  696.35238498  500.00000    500.0  500   562500.00
#[10,]  742.77587731  400.00000    400.0  400   640000.00
#[11,]  696.35238498  300.00000    300.0  300   562500.00
#[12,]  557.08190798  200.00000    200.0  200   360000.00
#[13,]  557.08190798  150.00000    150.0  150   360000.00
#[14,]  464.23492332  100.00000    100.0  100   250000.00
#[15,]  278.54095399   50.00000     50.0   50    90000.00

colSums(out[,1:3])
# NT-Allocation    Inequality      Equality
```

```
#      7246.800      7108.808      7246.800
```

`mosallocStepwiseFirst` *Multiobjective sample allocation for constraint multivariate and multidomain optimal allocation in survey sampling (a stepwise optimality procedure is processed first to force Pareto optimality of the solution)*

Description

Computes solutions to standard sample allocation problems under various precision and cost restrictions. The input data is transformed and parsed to the Embedded COnic Solver (ECOS) from the 'ECOSolveR' package. Multiple survey purposes are optimized simultaneously through a stepwise weighted Chebyshev minimization which forces Pareto optimality of solutions (cf. `mosalloc()`).

Usage

```
mosallocStepwiseFirst(
  D,
  d,
  A = NULL,
  a = NULL,
  C = NULL,
  c = NULL,
  l = 2,
  u = NULL,
 opts = list(sense = "max_precision", init_w = 1, mc_cores = 1L, max_iters = 100L)
)
```

Arguments

`D`	(type: matrix) The objective matrix. A matrix of either precision or cost units.
`d`	(type: vector) The objective vector. A vector of either fixed precision components (e.g. finite population corrections) or fixed costs.
`A`	(type: matrix) A matrix of precision units for precision constraints.
`a`	(type: vector) The right-hand side vector of the precision constraints.
`C`	(type: matrix) A matrix of cost coefficients for cost constraints
`c`	(type: vector) The right-hand side vector of the cost constraints.
`l`	(type: vector) A vector of lower box constraints.
`u`	(type: vector) A vector of upper box constraints.
`opts`	(type: list) The options used by the algorithms: `$sense` (type: character) Sense of optimization (default = `"max_precision"`, alternative `"min_cost"`). `$init_w` (type numeric or matrix) Preference weightings (default = 1; The weight

for first objective component must be 1).
`$mc_cores` (type: integer) The number of cores for parallelizing multiple input weightings stacked rowwise (default = 1L).
`max_iters` (type: integer) The maximum number of iterations (default = 100L).

Value

The function `mosallocStepwiseFirst()` returns a list containing the following components:

`$w` The initial preference weighting `opts$init_w`.

`$n` The vector of optimal sample sizes.

`$J` The optimal objective vector.

`$Objective` The objective value with respect to decision functional f. `NULL` if `opts$f = NULL`.

`$Utopian` Always `NULL` (consistency to `mosalloc()` output). `NULL` if `opts$f = NULL`.

`$Normal` The vector normal to the Pareto frontier at `$J`.

`$dfJ` Always `NULL` (consistency to `mosalloc()` output).

`$Sensitivity` The dual variables of the objectives and constraints.

`$Qbounds` The Quality bounds of the Lorentz cones.

`$Dbounds` The weighted objective constraints ($w * $J).

`$Scalepar` An internal scaling parameter.

`$Ecosolver` A list of ECOSolveR returns including:
`...$Ecoinfostring` The info string of `ECOSolveR::ECOS_csolve()`.
`...$Ecoredcodes` The redcodes of `ECOSolveR::ECOS_csolve()`.
`...$Ecosummary` Problem summary of `ECOSolveR::ECOS_csolve()`.

`$Timing` Run time info.

`$Iteration` Always `NULL` (consistency to `mosalloc()` output).

References

See:

Willems, F. (2025). A Framework for Multiobjective and Uncertain Resource Allocation Problems in Survey Sampling based on Conic Optimization (Doctoral dissertation). Trier University. doi:10.25353/ubtr9200484c5c89.

Examples

```
# Artificial population of 50 568 business establishments and 5 business
# sectors (data from Valliant, R., Dever, J. A., & Kreuter, F. (2013).
# Practical tools for designing and weighting survey samples. Springer.
# https://doi.org/10.1007/978-1-4614-6449-5, Example 5.2 pages 133-9)

# See also https://umd.app.box.com/s/9yvvibu4nz4q6rlw98ac/file/297813512360
# file: Code 5.3 constrOptim.example.R

Nh <- c(6221, 11738, 4333, 22809, 5467) # stratum sizes
ch <- c(120, 80, 80, 90, 150) # stratum-specific cost of surveying
```

```
# Revenues
mh.rev <- c(85, 11, 23, 17, 126) # mean revenue
Sh.rev <- c(170.0, 8.8, 23.0, 25.5, 315.0) # standard deviation revenue

# Employees
mh.emp <- c(511, 21, 70, 32, 157) # mean number of employees
Sh.emp <- c(255.50, 5.25, 35.00, 32.00, 471.00) # std. dev. employees

# Proportion of estabs claiming research credit
ph.rsch <- c(0.8, 0.2, 0.5, 0.3, 0.9)

# Proportion of estabs with offshore affiliates
ph.offsh <- c(0.06, 0.03, 0.03, 0.21, 0.77)

budget <- 300000 # overall available budget
n.min  <- 100 # minimum stratum-specific sample size

#-------------------------------------------------------------------------
# Problem: Minimization of the maximum relative variation of estimates for
# the total revenue, the number of employee, the number of businesses claimed
# research credit and the number of businesses with offshore affiliates
# subject to cost restrictions

l <- rep(n.min, 5) # minimum sample size ber stratum
u <- Nh            # maximum sample size per stratum
C <- rbind(ch, ch * c(-1, -1, -1, 0, 0))
c <- c(budget, - 0.5 * budget)
A <- NULL # no precision constraint
a <- NULL # no precision constraint

# Variance components for multidimensional objective
D <- rbind(Sh.rev**2 * Nh**2/sum(Nh * mh.rev)**2,
           Sh.emp**2 * Nh**2/sum(Nh * mh.emp)**2,
           ph.rsch * (1 - ph.rsch) * Nh**3/(Nh - 1)/sum(Nh * ph.rsch)**2,
           ph.offsh * (1 - ph.offsh) * Nh**3/(Nh - 1)/sum(Nh * ph.offsh)**2)

d <- as.vector(D %*% (1 / Nh)) # finite population correction

opts = list(sense = "max_precision",
            init_w = 1,
            mc_cores = 1L,
            max_iters = 100L)

res1 <- mosallocStepwiseFirst(D = D, d = d, C = C, c = c, l = l, u = u,
                              opts = opts)
w <- res1$J[1] / res1$J
w # [1] 1.000000 3.879692 2.653657 1.000000

opts = list(sense = "max_precision",
            init_w = w,
            mc_cores = 1L,
            max_iters = 100L)
```

```
res2 <- mosallocStepwiseFirst(D = D, d = d, C = C, c = c, l = l, u = u,
                              opts = opts)
res2$w # [1] 1.000000 3.879692 2.653657 1.000000

# Compare to function mosalloc (without stepwise procedure)
opts = list(sense = "max_precision",
            f = NULL, df = NULL, Hf = NULL,
            init_w = w,
            mc_cores = 1L, pm_tol = 1e-05,
            max_iters = 100L, print_pm = FALSE)
res3 <- mosalloc(D = D, d = d, C = C, c = c, l = l, u = u, opts = opts)

# Compare objectives
rbind(res1$J, res2$J, res3$J)
#            [,1]         [,2]         [,3]       [,4]
#[1,] 0.00170589 0.0004396972 0.0006428449 0.00170589
#[2,] 0.00170589 0.0004396971 0.0006428418 0.00170589
#[3,] 0.00170589 0.0004396971 0.0006428437 0.00170589

# Compare optimal sample sizes
rbind(res1$n, res2$n, res3$n)
#          [,1]     [,2]     [,3]     [,4]     [,5]
# [1,] 958.0503 290.7445 147.1801 602.8855 638.2687
# [2,] 958.0452 290.7445 147.1878 602.8847 638.2692
# [3,] 958.0483 290.7444 147.1832 602.8852 638.2689
```

`mosallocSTRS` *(Single-stage) stratified random sampling interface for functions* `mosalloc()` *and* `mosallocStepwiseFirst()`

Description

Interface for the functions `mosalloc()` and `mosallocStepwiseFirst()` of the same package which allows for a user-friendly data handling in the case of single-stage stratified random sampling.

Usage

```
mosallocSTRS(
  X_var,
  X_tot,
  N,
  listD,
  listA = NULL,
  listC = NULL,
  fpc = TRUE,
  l = 2,
  u = NULL,
```

```
  opts = list(sense = "max_precision", method = "WCM", f = NULL, df = NULL, Hf = NULL,
    init_w = 1, mc_cores = 1L, pm_tol = 1e-05, max_iters = 100L, print_pm = FALSE),
  ForceOptimality = FALSE,
  X_cost = NULL,
  X_fixed = NULL
)
```

Arguments

X_var	(type: matrix) A matrix containing stratum- (row) and variable- (column) specific precision units (e.g. variances).
X_tot	(type: matrix) A matrix containing stratum- (row) and variable- (column) specific totals.
N	(type: vector) A vector of stratum sizes.
listD	(type: list) A list of lists taking subpopulation- (domain/area) specific arguments. If opts$sense == "max_precision", elements are lists containing the following components which correspond to one specific precision target each: ..$stratum_id (type: numeric) A vector containing the indices of the strata considered for the current restriction. The indices must coincide with the row numbers of X_var and X_tot. ..$variate (type: character or numeric) The column name or column index of X_var to be addressed. ..$measure (type: character or numeric) A character ("relVAR" (relative variance) or "VAR" (variance)) or a numerical value between 0 and 1 indicates a gradation coefficient that balances between "relVar" (..$measure = 0) and "VAR" (..$measure = 1). ..$name (type: character) The name of the subpopulation (domain/area). If opts$sense == "min_cost", elements are lists containing the following components which correspond to one specific cost target each: ..$stratum_id (type: numeric) A vector containing the indices of the strata considered for the current objective. The indices must coincide with the row numbers of X_cost. ..$c_type (type: character or numeric) The column name or column index of X_cost to be addressed. ..$name (type: character) The name of the subpopulation (domain/area).
listA	(type: list) A list of lists taking subpopulation- (domain/area) specific arguments. Elements are lists containing the following components which in turn correspond to one specific precision restriction: ..$stratum_id (type: numeric) A vector containing the indices of the strata considered for the current restriction. The indices must coincide with the corresponding row numbers of X_var and X_tot. ..$variate (type: character or numeric) The column name or column index of X_var to be addressed. ..$measure (type: character) "RSE" (relative standard error) or "VAR" (variance). ..$bound (type: numeric) An upper bound to "RSE" (or "VAR"). ..$name (type: character) The name of the subpopulation (domain/area).

listC (type: list) A list of lists taking subpopulation- (domain/area) specific arguments. Elements are lists containing the following components which correspond to one specific cost restriction:
..$stratum_id (type: numeric) A vector containing the indices of the strata considered for the current restriction. The indices must coincide with the row numbers of X_var and X_tot.
..$c_coef (type: numeric) A vector of length length(stratum_id) containing the stratum-specific cost components for the set of strata that is going to be bounded by above and/or below.
..$c_upper (type: numeric) The cost upper bound value. NULL if not present.
..$c_lower (type: numeric) The cost lower bound value. NULL if not present.
..$name (type: character) The name of the subpopulation (domain/area).

fpc (type: logical) A TRUE or FALSE statement indicating whether the finite population correction should be considered.

l (type: vector) A vector of lower box constraints.

u (type: vector) A vector of upper box constraints.

opts (type: list) The options used by the algorithms:
$sense (type: character) Sense of optimization (default = "max_precision", alternative "min_cost").
$method (type: character) A character indicating scalarization method (default = "WCM", alternative "WSS"), $f must be NULL.
$f (type: function) Decision functional over the objective vector (default = NULL). The weighted Chebyshev minimization (WCM) minimizes for weighted maximum objective component. The weighted sum scalarization (WSS) minimizes for the weighted sum. For either case the weights are given by $init_w.
$df (type: function) The gradient of f (default = NULL).
$Hf (type: function) The Hesse matrix of f (default = NULL).
$init_w (type: numeric or matrix) Preference weightings (default = 1; The weight for first objective component must be 1).
$mc_cores (type: integer) The number of cores for parallelizing multiple input weightings stacked rowwise (default = 1L).
$pm_tol (type: numeric) The tolerance for the projection method (default = 1e-5).
max_iters (type: integer) The maximum number of iterations (default = 100L).
$print_pm (type: logical) A TRUE or FALSE statement whether iterations of the projection method should be printed (default = FALSE).

ForceOptimality (type: logical) A TRUE or FALSE statement indicating whether Pareto optimality should be ensured. Default is FALSE (for most practical problem formulations Pareto optimality is achieved by construction, e.g. in the case of one overall cost constraint). If TRUE, additional computation time is required. In this case, mosallocStepwiseFirst() is used internally.

X_cost (type: matrix) A matrix containing stratum- (row) and type- (column) specific cost coefficients associated with fixed cost. Types of cost might be, e.g. '$ US', 'minutes', 'sample size', etc. Default is NULL. The argument is required in the case of cost minimization (opt$sense == "min_cost").

X_fixed (type: matrix) A matrix containing stratum- (rows) and type- (columns) specific cost coefficients associated with fixed cost. Default is NULL. Used solely in the case of cost minimization (opt$sense == "min_cost").

Value

A mosaSTRS object or a list of mosaSTRS objects. A mosaSTRS object is a list containing the following components:

$sense Sense of optimization; max precision or min_cost.

$method The method used, either weighted sum scalarization (WSS) or weighted Chebyshev minimization (WCM).

$init_w The initial preference weightings (opts$init_w).

$opt_w The optimal weightings w.r.t. opts$init_w as opts$init_w might not lead to Pareto optimality. NULL if ForceOptimality = FALSE.

$n_opt The vector of optimal sample sizes.

$objective The objective values, including the sensitivity and the RSE.

$precision A data frame corresponding to the precision constraints.

$cost A data frame corresponding to the cost constraints.

$problem_components A list containing the input data to the optimization problem.

$output_mosalloc A list of function returns of mosalloc().

If opts$init_w has multiple rows, the function returns a list of mosaSTRS objects whose length equals the number of rows.

Examples

```
# Artificial population of 50 568 business establishments and 5 business
# sectors (data from Valliant, R., Dever, J. A., & Kreuter, F. (2013).
# Practical tools for designing and weighting survey samples. Springer.
# https://doi.org/10.1007/978-1-4614-6449-5, Example 5.2 pages 133-9)

# See also https://umd.app.box.com/s/9yvvibu4nz4q6rlw98ac/file/297813512360
# file: Code 5.3 constrOptim.example.R

Nh <- c(6221, 11738, 4333, 22809, 5467) # stratum sizes
ch <- c(120, 80, 80, 90, 150) # stratum-specific cost of surveying

# Revenues
mh.rev <- c(85, 11, 23, 17, 126) # mean revenue
Sh.rev <- c(170.0, 8.8, 23.0, 25.5, 315.0) # standard deviation revenue

# Employees
mh.emp <- c(511, 21, 70, 32, 157) # mean number of employees
Sh.emp <- c(255.50, 5.25, 35.00, 32.00, 471.00) # std. dev. employees

# Proportion of estabs claiming research credit
ph.rsch <- c(0.8, 0.2, 0.5, 0.3, 0.9)
```

```
# Proportion of estabs with offshore affiliates
ph.offsh <- c(0.06, 0.03, 0.03, 0.21, 0.77)

budget <- 300000 # overall available budget
n.min  <- 100 # minimum stratum-specific sample size

# Matrix containing stratum-specific variance components
X_var <- cbind(Sh.rev**2,
               Sh.emp**2,
               ph.rsch * (1 - ph.rsch) * Nh/(Nh - 1),
               ph.offsh * (1 - ph.offsh) * Nh/(Nh - 1))
colnames(X_var) <- c("rev", "emp", "rsch", "offsh")

# Matrix containing stratum-specific totals
X_tot <- cbind(mh.rev, mh.emp, ph.rsch, ph.offsh) * Nh
colnames(X_tot) <- c("rev", "emp", "rsch", "offsh")

# Examples
#-------------------------------------------------------------------------------
# Example 1: Univariate minimization of the variation of estimates for
# revenue subject to cost restrictions and precision restrictions to the
# relative standard error of estimates for the proportion of businesses with
# offshore affiliates. Additionally, there is one overall cost constraint and
# at least half of the provided budget must be spend to strata 1 to 3.

# Specify objectives via listD
listD <- list(list(stratum_id = 1:5, variate = "rev", measure = "relVAR",
                   name = "pop"))

# Specify precision constraints via listA
listA <- list(list(stratum_id = 1:5, variate = "offsh", measure = "RSE",
                   bound = 0.05, name = "pop"))

# Specify cost constraints via listC
listC <- list(list(stratum_id = 1:5, c_coef = ch, c_lower = NULL,
                   c_upper = budget, name = "Overall"),
              list(stratum_id = 1:3, c_coef = ch[1:3],
                   c_lower = 0.5 * budget, c_upper = NULL, name = "1to3"))

# Specify stratum-specific box constraints
l <- rep(n.min, 5) # minimum sample size per stratum
u <- Nh            # maximum sample size per stratum

# Specify parameter for mosalloc (method = "WSS")
opts <- list(sense = "max_precision",
             f = NULL, df = NULL, Hf = NULL,
             method = "WSS", init_w = 1,
             mc_cores = 1L, pm_tol = 1e-05,
             max_iters = 100L, print_pm = FALSE)

# Run mosallocSTRS with weighted sum scalarization (WSS)
resWSS  <- mosallocSTRS(X_var, X_tot, Nh, listD, listA, listC,
                            fpc = TRUE, l, u, opts)
```

```
summary(resWSS)

# Specify parameter for mosalloc (method = "WCM")
opts = list(sense = "max_precision",
            f = NULL, df = NULL, Hf = NULL,
            method = "WCM", init_w = 1,
            mc_cores = 1L, pm_tol = 1e-05,
            max_iters = 100L, print_pm = FALSE)

# Run mosallocSTRS with weighted Chebyshec minimization (WCM)
resWCM  <- mosallocSTRS(X_var, X_tot, Nh, listD, listA, listC,
                            fpc = TRUE, l, u, opts)

summary(resWCM)

# The optimal sample sizes vector can also be obtained by
summary(resWCM)$n_opt

# Hint: For univariate allocation problems 'WSS' and 'WCM' are equivalent!

#-------------------------------------------------------------------------
# Example 2: Minimization of the maximum relative variation of estimates for
# the total revenue, the number of employee, the number of businesses claimed
# research credit and the number of businesses with offshore affiliates
# subject to one overall cost constraint and at least half of the provided
# budget must be spend to strata 1 to 3.

# Specify objectives via listD
listD <- list(list(stratum_id = 1:5, variate = "rev", measure = "relVAR",
                   name = "pop"),
              list(stratum_id = 1:5, variate = "emp", measure = "relVAR",
                   name = "pop"),
              list(stratum_id = 1:5, variate = "rsch", measure = "relVAR",
                   name = "pop"),
              list(stratum_id = 1:5, variate = "offsh", measure = "relVAR",
                   name = "pop"))

# Specify cost constraints via listC
listC <- list(list(stratum_id = 1:5, c_coef = ch, c_lower = NULL,
                    c_upper = budget, name = "Overall"),
               list(stratum_id = 1:3, c_coef = ch[1:3],
                    c_lower = 0.5 * budget, c_upper = NULL, name = "1to3"))

# Specify stratum-specific box constraints
l <- rep(n.min, 5) # minimum sample size per stratum
u <- Nh            # maximum sample size per stratum

# Specify parameter for mosalloc (method = "WSS")
opts = list(sense = "max_precision",
            f = NULL, df = NULL, Hf = NULL,
            method = "WSS", init_w = 1,
            mc_cores = 1L, pm_tol = 1e-05,
```

```
            max_iters = 100L, print_pm = FALSE)

# Run mosallocSTRS with weighted sum scalarization (WSS)
resWSS  <- mosallocSTRS(X_var, X_tot, Nh, listD, NULL, listC,
                            fpc = TRUE, l, u, opts)

summary(resWSS)

# Specify parameter for mosalloc (method = "WCM")
opts = list(sense = "max_precision",
            f = NULL, df = NULL, Hf = NULL,
            method = "WCM", init_w = 1,
            mc_cores = 1L, pm_tol = 1e-05,
            max_iters = 100L, print_pm = FALSE)

# Run mosallocSTRS with weighted Chebyshec minimization (WCM)
resWCM  <- mosallocSTRS(X_var, X_tot, Nh, listD, NULL, listC,
                            fpc = TRUE, l, u, opts)

summary(resWCM)

# Since the WCM does not necessarily lead to Pareto optimal allocations,
# we might force this via a internal stepwise procedure by setting
# ForceOptimality = TRUE.

resWCM_FO  <- mosallocSTRS(X_var, X_tot, Nh, listD, NULL, listC,
                            fpc = TRUE, l, u, opts, ForceOptimality = TRUE)

summary(resWCM_FO)

#-------------------------------------------------------------------------------
# Example 3: Example 2 with multiple sets of preference weightings.

# Define a set of preference weightings, e.g.
w_1 <- c(1, 1, 1, 1)
w_2 <- c(1, 2, 2, 1)
w_3 <- c(1, 5, 5, 1)

# Combine the weightings to a matrix stacked rowwise
w <- rbind(w_1, w_2, w_3)

# Specify parameter for mosalloc() (method = "WCM"; not yet possible with WSS)
opts = list(sense = "max_precision",
            f = NULL, df = NULL, Hf = NULL,
            method = "WCM", init_w = w,
            mc_cores = 1L, pm_tol = 1e-05,
            max_iters = 100L, print_pm = FALSE)

# Run mosallocSTRS with weighted Chebyshec minimization (WCM)
res  <- mosallocSTRS(X_var, X_tot, Nh, listD, NULL, listC, fpc = TRUE,
                      l, u, opts)

summary(res)
```

```
#------------------------------------------------------------------------------
# Example 4: Minimization of survey cost subject to quality restrictions on
# subpopulation level.

X_cost <- matrix(ch, 5, 1, dimnames = list(1:5,"cost"))

# Specify cost objectives via listD
listD <- list(list(stratum_id = 1:5, c_type = "cost", name = "pop"))

# Specify quailty restrictions via listD. Here: 5 % relative standard error
listA <- list(list(stratum_id = 1:2, variate = "rev", measure = "RSE",
                   bound = 0.05, name = "S1-2"),
              list(stratum_id = 3:5, variate = "rev", measure = "RSE",
                   bound = 0.05, name = "S3-5"),
              list(stratum_id = 1:2, variate = "emp", measure = "RSE",
                   bound = 0.05, name = "S1-2"),
              list(stratum_id = 3:5, variate = "emp", measure = "RSE",
                   bound = 0.05, name = "S3-5"),
              list(stratum_id = 1:2, variate = "rsch", measure = "RSE",
                   bound = 0.05, name = "S1-2"),
              list(stratum_id = 3:5, variate = "rsch", measure = "RSE",
                   bound = 0.05, name = "S3-5"),
              list(stratum_id = 1:2, variate = "offsh", measure = "RSE",
                   bound = 0.05, name = "S1-2"),
              list(stratum_id = 3:5, variate = "offsh", measure = "RSE",
                   bound = 0.05, name = "S3-5"))

# Specify cost constraints
listC <- NULL

# Specify stratum-specific box constraints
l <- rep(n.min, 5) # minimum sample size per stratum
u <- Nh            # maximum sample size per stratum

# Specify parameters for mosalloc()
opts = list(sense = "min_cost",
            f = NULL, df = NULL, Hf = NULL,
            method = "WCM", init_w = 1,
            mc_cores = 1L, pm_tol = 1e-05,
            max_iters = 100L, print_pm = FALSE)

# Run mosallocSTRS()
res  <- mosallocSTRS(X_var, X_tot, Nh, listD, listA, NULL, fpc = TRUE,
                     l, u, opts, X_cost = X_cost)

summary(res)

# Optimal sample sizes
summary(res)$n_opt
```

print.summary.mosaSTRS *Print a summary.mosaSTRS object*

Description

Print-function for class `summary.mosaSTRS`.

Usage

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

Arguments

`x`	an object inheriting from class `summary.mosaSTRS`, representing the results of the function `mosallocSTRS()`
`...`	some methods for this generic require additional arguments. None are used in this method.

Value

Invisibly returns `x`.

summary.mosaSTRS *Summary a mosaSTRS object*

Description

Summary-function for class `mosaSTRS`

Usage

```
## S3 method for class 'mosaSTRS'
summary(object, ...)
```

Arguments

`object`	an object inheriting from class `mosaSTRS`, representing the results of the function `mosallocSTRS()`. This can also be a list of `mosaSTRS` objects.
`...`	some methods for this generic require additional arguments. None are used in this method.

Value

Either a `summary.mosaSTRS` object for a `mosaSTRS` object or a list of `summary.mosaSTRS` objects for a list of `mosaSTRS` objects. A `summary.mosaSTRS` object is a list containing the following components:

`$vname` Name of `object`.

`$sense` Sense of optimization; `max` `precision` or `min_cost`.

`$method` The method used weighted sum scalarization (WSS) or weighted Chebyshev minimization (WCM).

`$objout` A data frame corresponding to the objectives, including the values, the sensitivity, the weights and the RSE.

`$precision` A data frame corresponding to the precision constraints.

`$cost` A data frame corresponding to the cost constraints.

`$n_opt` A vector of optimal sample sizes w.r.t the `weights`.

Index