Package ‘stratification’

July 23, 2025

Type Package

Title Univariate Stratification of Survey Populations

Version 2.2-7

Date 2022-04-06

Author Louis-Paul Rivest [aut, cre],
Sophie Baillargeon [aut]

Maintainer Louis-Paul Rivest <Louis-Paul.Rivest@mat.ulaval.ca>

Description Univariate stratification of survey populations with a generalization of the Lavallee-Hidiroglou method of stratum construction. The generalized method takes into account a discrepancy between the stratification variable and the survey variable. The determination of the optimal boundaries also incorporate, if desired, an anticipated non-response, a take-all stratum for large units, a take-none stratum for small units, and a certainty stratum to ensure that some specific units are in the sample. The well known cumulative root frequency rule of Dalenius and Hodges and the geometric rule of Gunning and Horgan are also implemented.

Imports graphics, grDevices, stats, utils

LazyData true

License GPL-2

NeedsCompilation yes

Repository CRAN

Date/Publication 2022-04-06 21:00:02 UTC

Contents

`stratification-package`

Collection of Functions for Univariate Stratification of Survey Populations

---

Description

This package contains various functions for univariate stratification of survey populations. The well known cumulative root frequency rule by Dalenius and Hodges (1959) and the geometric rule by Gunning and Horgan (2004) are implemented. However, the main function implements a generalized Lavallee-Hidiroglou (1988) method of strata construction. It can be used with Sethi's (1963) or Kozak's (2004) algorithm. The generalized method takes into account a discrepancy between the stratification variable X and the survey variable Y. The method can consider a loglinear model with mortality between the variables (Baillargeon, Rivest and Ferland, 2007). When Kozak's algorithm is used, two additional models are available: a heteroscedastic linear model and a random replacement model as in Rivest (2002). The optimal boundaries determination also incorporates, if desired, an anticipated non-response, a take-all stratum for the large units and a take-none stratum for the small units. Moreover, units can be forced to be part of the sample by specifying a certainty stratum.

Details

Package:	stratification
Type:	Package
Version:	2.2-7
Date:	2022-04-06
License:	GPL-2

OVERWIEW OF THE FUNCTIONS

To determine the stratum sample sizes given a set of stratum boundaries: `strata.bh`

To determine the stratum boundaries first and, in a second step, the stratum sample sizes:
`strata.cumrootf`: cumulative root frequency method by Dalenius and Hodges (1959)
`strata.geo`: geometric method by Gunning and Horgan (2004)

To determine the optimal stratum boundaries and sample sizes in a single step:
`strata.LH`: generalized Lavallee-Hidiroglou method with Sethi's (1963) or Kozak's (2004) algorithm

All these functions create an object of class "strata", which can be visualized with the S3 methods `print.strata` and `plot.strata`. One can also apply, with the function `var.strata`, a stratified design to a survey variable Y different from the one used for the construction of the stratified design.

INFORMATION RELATIVE TO MANY FUNCTIONS

The functions `strata.bh`, `strata.cumrootf`, `strata.geo` and `strata.LH` need to be given:
`x`, the values of the stratification variable X,
`Ls`, the desired number of sampled strata,
`alloc`, an allocation rule, and
a target sample size `n` or a target level of precision `CV` for the survey estimator.
However, for Sethi's (1963) algorithm, only a target `CV` can be given. To reach a target `n` using the generalized Lavallee-Hidiroglou method, Kozak's (2004) algorithm has to be used with the `strata.LH` function.

TYPE OF STRATUM
In this package, four types of stratum exist: take-some, take-none, take-all and certainty. A take-some stratum is a stratum in which some units are sampled. A take-none stratum is a stratum for the smallest units in which no units are sampled. Its purpose is to ignore very small units. On the other hand, a take-all stratum is a stratum for the largest units in which every units are sampled. It allows to insure that the biggest units are in the sample. The following paragraph explains what the special stratum type called “certainty” is.

DEFINITION OF THE CERTAINTY STRATUM
It is possible to insure that some specific units are included in the sample with the argument `certain`. This argument is a vector containing the positions in the vector `x` of the units to be included with certainty in the sample. We say that these units form the certainty stratum. They are excluded from the population prior to the determination of the stratum boundaries, but they are accounted for in the calculation of the anticipated mean, the RRMSE, the total sample size and the optimization criteria. Essentially, these units form their own separate take-all stratum that is not subject to stratification. They do not have to be consecutive units according to the stratification variable, therefore their variance is meaningless. Non-response is not possible in the certainty stratum. The functions return a value named `certain.info` containing the number of units in the certainty stratum and their anticipated mean.

NUMBER OF STRATA
The `Ls` argument represents to the number of sampled strata. The term “sampled strata” refers to take-some and take-all strata only. Therefore, take-none and certain strata are not counted in `Ls`. If the stratified design does not have a take-none stratum then `Ls`=L is the total number of strata, otherwise `Ls`=$L-1$. In the total number of strata L, the certainty stratum, if any, is not counted since we do not need to find its boundaries.

STRATUM NUMBERING
Throughout the package, strata number 1 contains the smallest units and strata number L the biggest ones. So every vector of boundaries contains numbers in ascending order. The function `strata.bh` must be given boundaries `bh` fulfilling this condition. This remark also applies to the argument `initbh` of `strata.LH` used to give initial boundaries for the optimization algorithm. If a take-none stratum is requested, it is always the first one. On the other hand, if a take-none stratum is requested, it is always the last one.

DEFINITION OF STRATUM BOUNDARIES

Let's note $b_0, b_1, \ldots, b_L$ the stratum boundaries. Stratum h contains all the units with an X-value in the interval $[b_{h-1}, b_h)$ for $h = 1, \ldots, L$ such that $b_0 = min(X)$ and $b_L = max(X) + 1$, where $min(X)$ and $max(X)$ are respectively the minimum and the maximum values of the stratification variable. The argument `bh` of `strata.bh`, the argument `initbh` of `strata.LH` and the output value `bh` of any function of the package **stratification** with the prefix "strata" are length $L - 1$ vectors of the boundaries $b_1, b_2, \ldots, b_{L-1}$.

DETAILS ABOUT THE TAKE-NONE STRATUM

A non empty take-none stratum induces a bias in the estimator of the mean of Y, and the precision is measured by the relative root mean squared error (RRMSE), not by the coefficient of variation (CV). Regardless, in the functions the argument given to specify a target precision for the survey estimator is always named `CV`. However, in the output, the anticipated level of precision is named RRMSE for the functions accepting a `takenone` argument (`strata.bh` and `strata.LH`), and it is named CV for the other functions (`strata.cumrootf` and `strata.geo`).

When a `takenone` stratum is requested, one can specify a `bias.penalty` argument. We define the mean squared error for the estimator of the mean of Y by $MSE = (bias.penalty \times bias)^2 + variance$. It is sometimes possible to estimate the bias using the sum of the Y values in the take-none stratum from administrative data. In this situation, it might be appropriate to set `bias.penalty` to a value lower than 1. This will typically enlarge the take-none stratum. The value given to `bias.penalty` depends on the confidence level we have in the bias estimate. By default, it is assumed that no bias estimate is available and the whole bias contributes to the MSE (`bias.penalty=1`).

SPECIFICATION OF THE ALLOCATION RULE

The `alloc` argument must be a list containing the numeric objects `q1`, `q2` and `q3` which specify the allocation rule according to the general allocation scheme presented in Hidiroglou and Srinath (1993)

$$a_h = \frac{\gamma_h}{\sum_{\text{take-some}} \gamma_h} \qquad \text{where} \qquad \gamma_h = N_h^{2q_1} \bar{Y}_h^{2q_2} S_{yh}^{2q_3}.$$

Stratum sample sizes are calculated as :

$$n_{h\text{nonint}} = \begin{cases} 0 & \text{for take-none strata} \\ n \times a_h & \text{for take-some strata} \\ N_h & \text{for take-all strata} \end{cases}$$

A proportional allocation is obtained when `q1`=0.5 and `q2`=`q3`=0,
a power allocation is obtained when `q1`=`q2`=$p/2$ and `q3`=0, and
a Neyman allocation (the default) is obtained when `q1`=`q3`=0.5 and `q2`=0.

ROUNDING of the stratum sample sizes

Applying the allocation rule above gives real (non-integer) values for the sample sizes. These are named `nhnonint` in the package. The `nhnonint` values have to be rounded to get the integer sample sizes, named `nh` in the package. Here is how the rounding is done. If a target `CV` is requested, the values are simply rounded to the largest integer. However, if a target `n` is requested, the rounding is a little more complicated because the `nh` should sum to the target `n` and we do not want positive nh inferior to 1 to be rounded to zero. Therefore, we first round to 1 the positive nh inferior to 1. Then we calculate how many values (say nup) must be rounded to the largest integer and how many must be rounded to the smallest integer in order to fulfill the condition `sum(nh)=n`. We choose the nup values with the largest decimal part for the ceiling rounding, the other nh are rounded down.

ADJUSTMENT FOR A TAKE-ALL STRATUM
If, after applying the allocation rule, the stratified design contains at least one take-some stratum with ${n_h}_{\text{nonint}} > N_h$, the allocation is done again setting the take-some stratum with the largest units as a take-all stratum. This is done until ${n_h}_{\text{nonint}} \leq N_h$ for all the take-some strata or until there is only one take-some stratum left. This adjustment is done automatically throughout the package because the target n or CV might not be reached if one omits to do it. Only the function `strata.bh` allows not to do it (argument `takeall.adjust`).

Note: In special circumstances, the algorithm might result in more than one take-all stratum. If the non-response rate does not vary among the take-all strata, we can see them as forming one big take-all stratum. Otherwise, their boundaries influence the value of the optimization criteria (n or CV). So in the case of a varying non-response rate among the take-all strata, we cannot see them as forming one big take-all stratum.

SPECIFICATION OF A MODEL BETWEEN Y AND X
Every function can take into account a discrepancy between the stratification variable X and the survey variable Y. The functions `strata.bh`, `strata.cumrootf` and `strata.geo` perform allocation on the basis of anticipated moments whereas the `strata.LH` function goes further; it determines the optimal boundaries considering the anticipated moments. The following models for the relationship between Y and X can be specified through the `model` and `model.control` arguments:

- loglinear model with mortality (`model="loglinear"`):

$$Y = \begin{cases} \exp(\alpha + \texttt{beta}\ \log(X) + \texttt{epsilon}) & \text{with probability } p_h \\ 0 & \text{with probability } 1 - p_h \end{cases}$$

where `epsilon` $\sim N(0, \texttt{sig2})$ is independent of X. The parameter p_h is specified through `ph`, `ptakenone` and `pcertain` (elements of `model.control`). Note: The α parameter does not have to be specified because $exp(\alpha)$ is a multiplicative factor that has no impact on the outcome.

- heteroscedastic linear model (`model="linear"`):

$$Y = \texttt{beta}X + \texttt{epsilon}$$

where `epsilon` $\sim N(0, \texttt{sig2}\ X^{\texttt{gamma}})$.

- random replacement model (`model="random"`):

$$Y = \begin{cases} X & \text{with probability } 1 - \texttt{epsilon} \\ Xnew & \text{with probability } \texttt{epsilon} \end{cases}$$

where $Xnew$ is a random variable independent of X having the same distribution than X.

The `model.control` argument is a list that can supply any of the following model parameter:

`beta` A numeric: the slope of the "loglinear" or "linear" model. The default is 1.

`sig2` A numeric: the variance parameter of the "loglinear" or "linear" model. The default is 0.

`ph` A vector giving the survival rate in each of the `Ls` sampled strata for the "loglinear" model. A single number can be given if the rate doesn't vary among strata. The default is 1 in each stratum.

`ptakenone` A numeric: the survival rate in the take-none stratum, if a take-none stratum is added to the stratified design. The default is 1.

`pcertain` A numeric: the survival rate in the certainty stratum, if a certainty stratum is added to the stratified design. The default is 1.

`gamma` A numeric: the exponent of X in the residual variance of the "linear" model. The default is 0.

`epsilon` A numeric: the probability that the Y-value for a unit is equal to the X-value for a randomly selected unit in the population. It concerns the "random" model only. The default is 0.

Note: The default values of the parameters simplify any model to $Y = X$. Therefore, the default is always to consider that there is no discrepancy between the stratification and the survey variables. The `model` argument even has the default value `"none"`, which also means $Y = X$.

Author(s)

Sophie Baillargeon <Sophie.Baillargeon@mat.ulaval.ca> and
Louis-Paul Rivest <Louis-Paul.Rivest@mat.ulaval.ca>

References

Baillargeon, S., Rivest, L.-P., Ferland, M. (2007). Stratification en enquetes entreprises : Une revue et quelques avancees. *Proceedings of the Survey Methods Section, 2007 SSC Annual Meeting*.

Baillargeon, S. and Rivest, L.-P. (2009). A general algorithm for univariate stratification. *International Stratification Review*, **77**(3), 331-344.

Baillargeon, S. and Rivest L.-P. (2011). The construction of stratified designs in R with the package stratification. *Survey Methodology*, **37**(1), 53-65.

Dalenius, T. and Hodges, J.L., Jr. (1959). Minimum variance stratification. *Journal of the American Statistical Association*, **54**, 88-101.

Gunning, P. and Horgan, J.M. (2004). A new algorithm for the construction of stratum boundaries in skewed populations. *Survey Methodology*, **30**(2), 159-166.

Hidiroglou, M.A. and Srinath, K.P. (1993). Problems associated with designing subannual business surveys. *Journal of Business & Economic Statistics*, **11**, 397-405.

Kozak, M. (2004). Optimal stratification using random search method in agricultural surveys. *Statistics in Transition*, **6**(5), 797-806.

Lavallee, P. and Hidiroglou, M.A. (1988). On the stratification of skewed populations. *Survey Methodology*, **14**, 33-43.

Rivest, L.-P. (2002). A generalization of the Lavallee and Hidiroglou algorithm for stratification in business surveys. *Survey Methodology*, **28**(2), 191-198.

Sethi, V. K. (1963). A note on optimum stratification of populations for estimating the population means. *The Australian Journal of Statistics*, **5**, 20-33.

`CochranHorganData` *Populations Analyzed in Gunning and Horgan (2004) and Cochran (1961)*

Description

The first population `Debtors` is an accounting population of debtors in an Irish firm, detailed in Horgan (2003). The other populations are three of the skewed populations in Cochran (1961). These are:
`UScities`: the population in thousands of US cities in 1940;
`UScolleges`: the number of students in four-year US colleges in 1952-1953;
`USbanks`: the resources in millions of dollars of large commercial US banks.

Usage

```
Debtors
UScities
UScolleges
USbanks
```

Format

The formats of these data sets are, respectively:
num [1:3369] 40 40 40 40 40 40 40 40 40 40 ...
num [1:1038] 10 10 10 10 10 10 10 10 10 10 ...
num [1:677] 200 201 202 202 207 210 211 213 215 217 ...
num [1:357] 70 71 72 72 72 73 73 73 73 73 ...

Source

Jane M. Horgan

References

Cochran, W.G. (1961). Comparison of methods for determining stratum boundaries. *Bulletin of the International Statistical Institute*, **32**(2), 345-358.

Gunning, P. and Horgan, J.M. (2004). A new algorithm for the construction of stratum boundaries in skewed populations. *Survey Methodology*, **30**(2), 159-166.

Horgan, J.M. (2003). A list sequential sampling scheme with applications in financial auditing. *IMA Journal of Management Mathematics*, **14**, 1-18.

Examples

```
### Reproduction of the results in Table 4 and Table 7 part 3 (case L=5) of
### Gunning and Horgan (2004). The differences in the nh come from different
### rounding. The more important differences observed for the cumulative
### root frequency method are due to the use of different numbers of classes.
strata.geo(x=Debtors, n=100, Ls=5, alloc=c(0.5,0,0.5))
strata.cumrootf(x=Debtors, n=100, Ls=5, alloc=c(0.5,0,0.5), nclass=40)
strata.LH(x=Debtors, CV=0.0360, Ls=5, alloc=c(0.5,0,0.5), takeall=1, algo="Sethi")

strata.geo(x=UScities, n=100, Ls=5, alloc=c(0.5,0,0.5))
strata.cumrootf(x=UScities, n=100, Ls=5, alloc=c(0.5,0,0.5), nclass=40)
strata.LH(x=UScities, CV=0.0144, Ls=5, alloc=c(0.5,0,0.5), takeall=1, algo="Sethi")

strata.geo(x=UScolleges, n=100, Ls=5, alloc=c(0.5,0,0.5))
strata.cumrootf(x=UScolleges, n=100, Ls=5, alloc=c(0.5,0,0.5), nclass=40)
strata.LH(x=UScolleges, CV=0.0184, Ls=5, alloc=c(0.5,0,0.5), takeall=1, algo="Sethi")

strata.geo(x=USbanks, n=100, Ls=5, alloc=c(0.5,0,0.5))
strata.cumrootf(x=USbanks, n=100, Ls=5, alloc=c(0.5,0,0.5), nclass=40)
strata.LH(x=USbanks, CV=0.0110, Ls=5, alloc=c(0.5,0,0.5), takeall=1, algo="Sethi")
```

MRTS — *Simulated Data from the Monthly Retail Trade Survey (MRTS) of Statistics Canada*

Description

This data set is a vector containing the simulated values of a realistic stratification variable: the size measure used for Canadian retailers in the Monthly Retail Trade Survey (MRTS) carried out by Statistics Canada. This size measure is created using a combination of independent survey data and three administrative variables from the corporation tax return. The MRTS aims at estimating sales from retailers, which are a key monthly indicator of consumer purchasing patterns in Canada.

Usage

```
MRTS
```

Format

The format is: num [1:2000] 141 209 238 257 261 ...

Source

The data set has been simulated with the command:
`exp(rst(n=2000, location=9.7093354, scale=0.7885551,`
`shape=-0.6384867, df=5.6243544))`.
The function `rst` comes from the package **sn**. It generates random numbers for the skew-t distribution. The parameters of the distribution have been estimated with the assistance of Michel Ferland from Statistics Canada to be representative of the measure of size used in the MRTS.

References

Baillargeon, S., Rivest, L.-P., Ferland, M. (2007). Stratification en enquetes entreprises : Une revue et quelques avancees. *Proceedings of the Survey Methods Section, 2007 SSC Annual Meeting*.

Examples

```
# Production of results similar to those in Table 1 of Baillargeon, Rivest
# and Ferland (2007). The results are not the same because calculations in
# the paper were conducted on real data whereas, for confidentiality reason,
# the MRTS data included in the package is simulated.
geo <- strata.geo(x=MRTS, CV=0.01, Ls=4, alloc=c(0.5,0,0.5))
geo
aRRMSE.geo <- var.strata(geo, model="loglinear",
              model.control=list(beta=0.9, sig2=0.015, ph=c(0.8,0.9,0.95,1)))
aRRMSE.geo$RRMSE
plot(geo, logscale=TRUE)
# The geometric method does not perform well because of some small units

cumrootf <- strata.cumrootf(x=MRTS, nclass=500, CV=0.01, Ls=4, alloc=c(0.5,0,0.5))
cumrootf
aRRMSE.cum <- var.strata(cumrootf, rh=c(0.85,0.9,0.9,1), model="loglinear",
              model.control=list(beta=0.9, sig2=0.015, ph=c(0.8,0.9,0.95,1)))
aRRMSE.cum$RRMSE

LH <- strata.LH(x=MRTS, CV=0.01, Ls=4, alloc=c(0.5,0,0.5), takeall=1, algo="Sethi")
LH
aRRMSE.LH <- var.strata(LH, rh=c(0.85,0.9,0.9,1), model="loglinear",
             model.control=list(beta=0.9, sig2=0.015, ph=c(0.8,0.9,0.95,1)))
aRRMSE.LH$RRMSE

LH.full <- strata.LH(x=MRTS, CV=0.01, Ls=4, alloc=c(0.5,0,0.5), takeall=1,
           algo="Sethi", rh=c(0.85,0.9,0.9,1), model="loglinear",
           model.control=list(beta=0.9, sig2=0.015, ph=c(0.8,0.9,0.95,1)))
LH.full
aRRMSE.LH.full <- var.strata(LH.full, rh=c(0.85,0.9,0.9,1), model="loglinear",
                  model.control=list(beta=0.9, sig2=0.015, ph=c(0.8,0.9,0.95,1)))
aRRMSE.LH.full$RRMSE
```

SHS — *Data from the 2001 Survey of Household Spending (SHS), Statistics Canada*

Description

This data set contains some variables from the 2001 Survey of Household Spending (SHS) carried out by Statistics Canada. The main purpose of this survey is to obtain detailed information about household spending during the reference year.

Usage

```
SHS
```

Format

A data frame with 16057 observations on the following 7 variables.

`CASEID`	Identification number
`WEIGHT`	Weight at household level
`PROVINCP`	Province or territory code
`URBRUR`	Urban rural code
`URBSIZEP`	Size of area of residence code
`HHINCTOT`	Household income before taxes
`M101`	Household spending on recreation

Details

In this package, `HHINCTOT` is used as a stratification variable and `M101` as a survey variable.

Source

Income Statistics Division, Statistics Canada.

Examples

```
# Estimation of the response rate
X <- SHS$HHINCTOT[SHS$HHINCTOT>0]
Y <- SHS$M101[SHS$HHINCTOT>0]
Y[Y<0] <- 0
p<-sum(Y>0)/length(Y)

# Study of the relationship between X and Y for the active units
Xactive <- SHS$HHINCTOT[SHS$HHINCTOT>0&SHS$M101>0]
Yactive <- SHS$M101[SHS$HHINCTOT>0&SHS$M101>0]
plot(log(Xactive), log(Yactive))
# Extreme values are omitted for a more robust estimation
keep <- Xactive/Yactive>quantile(Xactive/Yactive,0.01)&
        Xactive/Yactive<quantile(Xactive/Yactive,0.99)
plot(log(Xactive)[keep], log(Yactive)[keep])
reg <- lm( log(Yactive)[keep]~log(Xactive)[keep] )
summary(reg)

# Stratification assuming X=Y
nomodel <- strata.LH(x=X, CV=0.05, Ls=3, alloc=c(0.5,0,0.5), takeall=0,
           model="none", algo.control=list(trymany=FALSE, rep=2))
nomodel
var.strata(nomodel, y=Y) # The target CV is far from being reached

# Stratification taking into account a loglinear model with mortality
# between X and Y, using the estimated parameters values
model <- strata.LH(x=X, CV=0.05, Ls=3, alloc=c(0.5,0,0.5), takeall=0,
         model="loglinear", model.control=list(beta=reg$coef[2],
         sig2=summary(reg)$sigma^2, ph=0.97), initbh=nomodel$bh,
         algo.control=list(trymany=FALSE, rep=2))
```

```
model
var.strata(model,y=Y) # The target CV is reached
```

strata.bh — *Stratification of a Population Given a Set of Boundaries*

Description

The function `strata.bh` stratifies a population given a set of boundaries. It calculates the stratum sample sizes and the anticipated coefficient of variation or relative root mean squared error.

Usage

```
strata.bh(x, bh, n = NULL, CV = NULL, Ls = 3, certain = NULL,
          alloc = list(q1 = 0.5, q2 = 0, q3 = 0.5), takenone = 0,
          bias.penalty = 1, takeall = 0, takeall.adjust = TRUE,
          rh = rep(1, Ls), model = c("none", "loglinear", "linear",
          "random"), model.control = list())
```

Arguments

`x`	A vector containing the values of the stratification variable X for every unit in the population.
`bh`	A vector of the $L-1$ stratum boundaries $(b_1, b_2, \ldots, b_{L-1})$ where L is the total number of strata (excluding the certainty stratum, if any). Therefore, if `takenone=0` then L=`Ls`, and if `takenone=1` then L=`Ls`+1.
`n`	A numeric: the target sample size. It has no default value. The argument `n` or the argument `CV` must be input.
`CV`	A numeric: the target coefficient of variation or relative root mean squared error if `takenone`=1. It has no default value. The argument `CV` or the argument `n` must be input.
`Ls`	A numeric: the number of sampled strata (take-none and certain strata are not counted in `Ls`). The default is 3.
`certain`	A vector giving the position, in the vector `x`, of the units that must be included in the sample (see `stratification-package`). By default `certain` is `NULL`, which means that no units are a priori chosen to be in the sample.
`alloc`	A list specifying the allocation scheme. The list must contain 3 numerics for the 3 exponents `q1`, `q2` and `q3` in the general allocation scheme (see `stratification-package`). The default is Neyman allocation (`q1`=`q3`=0.5 and `q2`=0)
`takenone`	A numeric: the number of take-none strata (0 or 1). The default is 0, i.e. no take-none stratum is included.
`bias.penalty`	A numeric between 0 and 1 giving the penalty for the bias in the anticipated mean squared error (MSE) of the survey estimator (see `stratification-package`). This argument is relevant only if `takenone`=1. The default is 1.

`takeall`	A numeric: the number of take-all strata (one of {0, 1, ..., `Ls`-1}). The default is 0, i.e. no take-all stratum is included.
`takeall.adjust`	A logical. If `TRUE` (the default), when $n_h > N_h$ for a take-some stratum, the `takeall` argument is increased by one and the allocation is carried out again. This is done as long as $n_h \leq N_h$ for every take-some stratum. If `FALSE`, no adjustment is made. Note: in other functions of the package **stratification**, this adjustment is not optional; it is made automatically (see `stratification-package`).
`rh`	A vector giving the anticipated response rates in each of the `Ls` sampled strata. A single number can be given if the rates do not vary among strata. The default is 1 in each stratum.
`model`	A character string identifying the model used to describe the discrepancy between the stratification variable X and the survey variable Y. It can be `"none"` if one assumes $Y = X$, `"loglinear"` for the loglinear model with mortality, `"linear"` for the heteroscedastic linear model or `"random"` for the random replacement model (see `stratification-package` for a description of these models). The default is `"none"`.
`model.control`	A list of model parameters (see `stratification-package`). The default values of the parameters correspond to the model $Y = X$.

Value

`Nh`	A vector of length L containing the population sizes N_h, i.e. the number of units in each stratum.
`nh`	A vector of length L containing the sample sizes n_h, i.e. the number of units to sample in each stratum. See `stratification-package` for information about the rounding used to get these integer values.
`n`	The total sample size (`sum(nh)`).
`nhnonint`	A vector of length L containing the non-integer values of the sample sizes, obtained directly from applying the allocation rule (see `stratification-package`).
`certain.info`	A vector giving statistics for the certainty stratum (see `stratification-package`). It contains `Nc`, the number of units chosen a priori to be in the sample, and `meanc`, the anticipated mean of Y for these units.
`opti.nh`	The final value of the criteria to optimize (either the total sample size n if a target `CV` was given or the RRMSE if a target `n` was given) calculated with the integer stratum sample sizes `nh`.
`opti.nhnonint`	The final value of the criteria to optimize (either the total sample size n if a target `CV` was given or the RRMSE if a target `n` was given) calculated with the non-integer stratum sample sizes `nhnonint`.
`meanh`	A vector of length L containing the anticipated means of Y in each stratum.
`varh`	A vector of length L containing the anticipated variances of Y in each stratum.
`mean`	A numeric: the anticipated global mean value of Y.
`RMSE`	A numeric: the root mean squared error (or standard error if `takenone`=0) of the anticipated global mean of Y. This is defined as the squared root of: (`bias.penalty` x bias of the mean)^2 + variance of the mean.

`RRMSE`	A numeric: the anticipated relative root mean squared error (or coefficient of variation if `takenone`=0) for the mean of Y, i.e. `RMSE` divided by `mean`.
`relativebias`	A numeric: the anticipated relative bias of the estimator, i.e. (`bias.penalty` x bias of the mean) divided by `mean`. If `takenone`=0, this numeric is zero.
`propbiasMSE`	A numeric: the proportion of the MSE attributable to the bias of the estimator, i.e. (`bias.penalty` x bias of the mean)^2 divided by the MSE of the `mean`. If `takenone`=0, this numeric is zero.
`stratumID`	A factor, having the same length as the input `x`, which values are either 1, 2, ..., L or `"certain"`. The value `"certain"` is given to units a priori chosen to be in the sample. This factor identifies, for each observation, the stratum to which it has been assigned.
`takeall`	The number of take-all strata in the final solution. Note: It is possible that $n_h = N_h$ for non take-all strata because the condition for an automatic addition of a take-all stratum is $n_h > N_h$.
`call`	The function call (object of class "call").
`date`	A character string that contains the system date and time when the function ended.
`args`	A list of all the argument values input to the function or set by default.

Author(s)

Sophie Baillargeon <`Sophie.Baillargeon@mat.ulaval.ca`> and
Louis-Paul Rivest <`Louis-Paul.Rivest@mat.ulaval.ca`>

References

Baillargeon, S. and Rivest L.-P. (2011). The construction of stratified designs in R with the package stratification. *Survey Methodology*, **37**(1), 53-65.

See Also

`print.strata`, `plot.strata`, `strata.cumrootf`, `strata.geo`, `strata.LH`

Examples

```
adjust <- strata.geo(x=USbanks, CV=0.01, Ls=4, alloc=c(0.35,0.35,0))
adjust
adjust$nhnonint
noadjust <- strata.bh(x=USbanks, bh=adjust$bh, CV=0.01, Ls=4,
            alloc=c(0.35,0.35,0), takeall=0, takeall.adjust=FALSE)
noadjust
noadjust$nhnonint
# without the adjustment for a take-all stratum, n is smaller than
# with the adjustment, but the target CV is not reached.
```

strata.LH *Generalized Lavallee-Hidiroglou Method of Strata Construction*

Description

This function aims at constructing optimal strata with a generalized Lavallee-Hidiroglou (1998) method. The function uses Kozak's (2004) algorithm by default, but it can also apply Sethi's (1963) algorithm (argument `algo="Sethi"`). The function can take into account a discrepancy between the stratification variable X and the survey variable Y. It can consider a loglinear model with mortality between the variables (Baillargeon and Rivest, 2009). With Kozak's algorithm, two additional models are implemented: an heteroscedastic linear model and a random replacement model as in Rivest (2002). The determination of the optimal boundaries also incorporates, if desired, an anticipated non-response, a take-all stratum for the large units, a take-none stratum for the small units, and a certainty stratum to ensure that some specific units are in the sample.

Sethi's algorithm is not used by default because it can be numerically unstable, especially with a take-none stratum. Better results were obtained with Kozak's algorithm in our numerical experiments.

Usage

```
strata.LH(x, n = NULL, CV = NULL, Ls = 3, certain = NULL,
          alloc = list(q1 = 0.5, q2 = 0, q3 = 0.5), takenone = 0,
          bias.penalty = 1, takeall = 0, rh = rep(1, Ls),
          model = c("none", "loglinear", "linear", "random"),
          model.control = list(), initbh = NULL,
          algo = c("Kozak", "Sethi"), algo.control = list())
```

Arguments

`x`	A vector containing the values of the stratification variable X for every unit in the population.
`n`	A numeric: the target sample size. It has no default value. The argument `n` or the argument `CV` must be input.
`CV`	A numeric: the target coefficient of variation or relative root mean squared error if `takenone`=1. It has no default value. The argument `CV` or the argument `n` must be input.
`Ls`	A numeric: the number of sampled strata (take-none and certain strata are not counted in `Ls`). The default is 3.
`certain`	A vector giving the position, in the vector `x`, of the units that must be included in the sample (see `stratification-package`). By default `certain` is `NULL`, which means that no units are chosen a priori to be in the sample.
`alloc`	A list specifying the allocation scheme. The list must contain 3 numerics for the 3 exponents `q1`, `q2` and `q3` in the general allocation scheme (see `stratification-package`). The default is Neyman allocation (`q1`=`q3`=0.5 and `q2`=0)

`takenone`	A numeric: the number of take-none strata (0 or 1). The default is 0, i.e. no take-none stratum is included.
`bias.penalty`	A numeric between 0 and 1 giving the penalty for the bias in the anticipated mean squared error (MSE) of the survey estimator (see `stratification-package`). This argument is relevant only if `takenone=1`. The default is 1.
`takeall`	A numeric: the number of take-all strata (one of {0, 1, ..., `Ls-1`}). The default is 0, i.e. no take-all stratum is included.
`rh`	A vector giving the anticipated response rates in each of the `Ls` sampled strata. A single number can be given if the rates do not vary among strata. The default is 1 in each stratum.
`model`	A character string identifying the model used to describe the discrepancy between the stratification variable X and the survey variable Y. It can be `"none"` if one assumes $Y = X$, `"loglinear"` for the loglinear model with mortality, `"linear"` for the heteroscedastic linear model or `"random"` for the random replacement model (see `stratification-package` for a description of these models). The default is `"none"`, so the original Lavallee-Hidiroglou (1998) method of strata construction is used. The last two models `"linear"` and `"random"` are only available with Kozak's algorithm.
`model.control`	A list of model parameters (see `stratification-package`). The default values of the parameters correspond to the model $Y = X$.
`initbh`	A vector of initial stratum boundaries (see **Details**).
`algo`	A character string identifying which optimization algorithm is to be used. It can be "Kozak" for Kozak's (2004) algorithm or "Sethi" for Sethi's (1963) algorithm (see **Details**). The default is "Kozak" because it performs better and offers more options than Sethi's algorithm.
`algo.control`	A list of parameters to control the optimization algorithm (see **Details**).

Details

CALCULATION OF THE OPTIMIZATION CRITERIA

Kozak's and Sethi's algorithms aim at finding optimal stratum boundaries. This optimality refers to a criteria :
the total sample size n if a target `CV` was given
the RRMSE if a target `n` was given (not available for Sethi's algorithm).
This criteria, no matter if it is n or the RRMSE, is a function of the stratum sample sizes, among others. But these stratum sample sizes can be non-integer (`nhnonint` obtained directly from applying the allocation rule) or integer (`nh` obtained from rounding the `nhnonint`), as mentioned in `stratification-package`. The optimization criteria does not vary much depending on the sample size used, but it is not exactly the same. Let's call `opti.nh` the criteria calculated with `nh` and `opti.nhnonint` the criteria calculated with `nhnonint`. Optimizing based on `opti.nh` vs `opti.nhnonint` might not give the same result. In fact, according to our numerical tests, it is a little better to optimize based on `opti.nh`. This is logical since the sample sizes used in practice, to apply a stratified design, are the integer one. Therefore, as of version 2.1-0 of the package, Kozak's algorithm uses by default the integer sample sizes `nh` to calculate the optimization criteria (but it can also use `opti.nhnonint` by setting `idopti="nhnonint"` in `algo.control`). However, Sethi's

algorithm works with derivatives to perform optimization. Therefore, it does not treat the optimization as a discrete problem as Kozak's algorithm does. It can only work with a real optimization criteria, i.e. the criteria `opti.nhnonint` calculated with non-integer sample size. This is one more reason to favour Kozak's algorithm over Sethi's one.

INITIAL STRATUM BOUNDARIES

Both Kozak's and Sethi's algorithm are iterative and they need starting values. It can be given by the user with the `initbh` argument. If not given, the default initial boundaries are obtained with the function `strata.cumrootf` (cumulative root frequency method) for Kozak's algorithm. For Sethi's algorithm, the default value of `initbh` is the set of `Ls`-1 equidistant points along the range of the X-values (arithmetic starting points of Gunning and Horgan (2007)). Let's define $d = (max(X) - min(X))/Ls$, the arithmetic boundaries are $b_h = min(X) + h \times d$ for $h = 1, \ldots, Ls - 1$.

The default initial boundaries are not the same for the two algorithms because Kozak's algorithm uses by default the criteria `opti.nh` whereas Sethi's algorithm can only use `opti.nhnonint`. In our numerical experiments with the criteria `opti.nh`, the cumulative root frequency boundaries performed a little better than geometric, arithmetic or robust boundaries (for a definition of theses boundaries, see details about the `trymany` algorithm parameter). On the other hand, the arithmetic boundaries had good performances (Baillargeon and Rivest 2009) with the criteria `opti.nhnonint`. Let's note however that we never found, in any of our numerical experiments, initial boundaries clearly better than others.

The length of the vector `initbh` can be `Ls`-1 or L-1, where L is the total number of strata. When `takenone=0`, L=`Ls` and it does not make any difference. But for a stratified design with a take-none stratum, it means that one can give as `initbh` argument the vector of initial boundaries $(b_1, b_2, \ldots, b_{L-1})$ or $(b_2, \ldots, b_{L-1})$. In the second option, the upper boundary of the take-none stratum is not given. In that case, it is set by default to the first percentile of `x`. If this first percentile is equal to the minimum value of `x`, this initial boundary would lead to an empty take-none stratum. In that case, the initial boundary of the take-none stratum is rather set to the second smallest value of `x`.

Sometimes, the specified initial boundaries are not suitable for Kozak's algorithm. As written below, this algorithm verifies at each iteration that the sampled strata contains at least `minNh` units and that they have positive sample sizes `nh`. If the initial boundaries do not meet these conditions, the initial optimization criteria is not comparable to the criteria for boundaries respecting the conditions. Therefore, in such a situation, a warning is produced and the algorithm is not run.

ALGORITHMS

Sethi: The formulas implemented for Sethi's algorithm are presented in Baillargeon, Rivest and Ferland (2007). As mentioned previously, this algorithm is available for a target CV only. Moreover, it works with derivatives to perform optimization. Therefore, it can only work with a real optimization criteria (`opti.nhnonint`).

Kozak: Kozak's algorithm is described in Kozak (2004). It starts at the initial boundaries `initbh` and at each iteration, it chooses a stratum boundary at random and a random modification for this boundary among the 2*`maxstep` possible alternatives. If this modification reduces the optimization criteria, creates sampled strata containing at least `minNh` units and leads to positive `nh`, it is accepted.

Otherwise, the boundaries are not changed. If the boundaries remain unchanged for `maxstill` consecutive iterations, the algorithm has reached convergence and it stops.
Kozak's (2004) used the condition `nh` >= 2 instead of `nh` > 0. We chose to modify this requirement because we noticed that more severe conditions sometimes prevented the algorithm from selecting a path leading to the optimal solution. According to our numerical experiments, if one sets a posteriori to 2 the `nh`'s that are equal to 1 at the end of the algorithm, the new sample size is smaller than or equal to the one obtained with the condition `nh` >= 2.
If no `initbh` argument is given, the algorithm is run with three different sets of initial boundaries (see details of the algorithm parameter `trymany`). If `initbh` is given, it uses these initial boundaries. Bot no matter how many initial boundaries are tried, the algorithm is run `rep` times with each. Running the algorithm many times helps having more stable results. Since the algorithm is random, two runs of the algorithm, event from the same starting point, can lead to different results. Setting `rep` to a large value and trying more than one set of initial boundaries increases the calculation time, but also increases the chance of finding the global minimum. also helps in finding the global minimum.

The optimization of stratum boundaries given a target CV or a target n is a discrete problem. The number of possible sets of boundaries is `choose(Nu-1,L-1)` when `takenone=0`, and `choose(Nu,L-1)` when `takenone=1`, where `Nu` stands for the number of unique values in the `x`-vector from which units in the certainty stratum, if any, heve been removed. When this number is not too large, trying them all is numerically possible. That's what the function `strata.LH` do, for Kozak's algorithm, when the number of possible solutions is lower than the `minsol` parameter in `algo.control`. This complete enumeration of all possible cases ensure that the global minimum is reached.

Note: Since version 2.2-0 of the package **stratification**, the modified Kozak's algorithm, which is a non-random version of Kozak's algorithm, is no more available because it never performed as well as the original Kozak's algorithm in our numerical experiments.

ALGORITHM PARAMETERS

The `algo.control` argument is a list to supply any of the following parameters which control the algorithm. Sethi's algorithm only uses the first argument `maxiter`.

`maxiter` A numeric: the maximal number of iterations. The default is 500 for Sethi's algorithm and 10 000 for Kozak's. It is only used to prevent from infinite loops in case of non-convergence.

`minsol` A numeric for Kozak's algorithm only: if the number of possible sets of boundaries is lower than `minsol`, a complete enumeration of the solutions is performed instead of running the algorithm. Every set of boundaries is tried, which ensures that the global minimum is reached. The default value of `minsol` is 10 000. This parameter has to take a value between 2 and 2 000 000.

`idopti` A character string for Kozak's algorithm only: this argument determines which stratum sample sizes are going to be used to calculate the optimization criteria. It can take the value `"nh"` (criteria calculated with the integer sample sizes `nh`) or `"nhnonint"` (criteria calculated with the non-integer sample sizes `nhnonint`). The default value is `"nh"` since it gives slightly better results than `idopti="nhnonint"` and also because the integer sample sizes are the ones used in practice. When a complete enumeration is performed, the criteria is automatically calculated with integer stratum sample sizes. Note: Prior to version 2.1-0 of the package **stratification**, only the option `idopti="nhnonint"` was available (in fact the algorithm parameter `idopti` did not exist).

`minNh` A numeric for Kozak's algorithm only: the minimum number of units required in each sampled stratum (no restriction is put on the take-none stratum, if included). `minNh` must be greater or equal to 2, which is the default.

`maxstep` A numeric for Kozak's algorithm only: the maximal step for boundary modification (see the algorithm description above). The default is `Nu/10`, rounded up and truncated to 100 (`Nu` is the number of unique values in the `x`-vector from which units in the certainty stratum, if any, heve been removed). This default value is, for most populations, much larger than the initial suggestion by Kozak (2004) of a integer no bigger than 5. Our numerical experiments showed us that a `maxstep` value of about 3 (package **stratification** initial default value) is adequate when the optimization criteria is calculated with non-integer stratum sample sizes (`opti.nhnonint`). However, when this criteria is calculated with integer stratum sample sizes (`opti.nh`), the algorithm never do worst and sometimes reaches an even lower optimization criteria with a larger `maxstep`. The parameter `maxstill` must be increased consequently (see below). The downside of large `maxstep` and `maxstill` values is that many iterations are needed for the algorithm to converge, making it longer to run. We implemented a solution to this problem. This solution is based on the fact that when the algorithm is close to the solution, it can only accept boundaries modifications with small steps. The large steps are useful during the first iterations of the algorithm to let the boundaries move freely, but it becomes useless when convergence is close. Therefore, the values of `maxstep` and `maxstill` are brought down to 3 and 50, respectively, when new boundaries are accepted under these circumstances : the step in the boundary modification is lower than 3 in absolute value, more than 50 iterations without change happened before this change, the relative change in the optimization criteria is small. This rule is arbitrary, but it seems to work well so far. It gives results as good as without changing the values of `maxstep` and `maxstill`, but it runs much faster because it needs less iterations to converge.

`maxstill` A numeric for Kozak's algorithm only: the maximal number of iterations without a change in the boundaries (see the algorithm description above). The default is `maxstep*10`, bounded between 50 and 500. It depends on the value of `maxstep` because we increased the default value of this parameter by making it a function of the number of unique values in the `x`-vector (see above). The upper bound of 500 ensures that the algorithm does not have to make too much iterations (which take time) to converge.

`rep` A numeric for Kozak's algorithm only: the number of repetitions of the algorithm (see the algorithm description above). The default is 5. Since version 2.2-0 of the package **stratification**, the option `rep="change"` has been replaced by the option `trymany=TRUE`.

`trymany` A logical for Kozak's algorithm only: if `trymany=TRUE`, three sets of initial boundaries are tried instead of one. These boundaries are cumulative root frequency (Dalenius and Hodges, 1959), geometric (Gunning and Horgan, 2004) and robust ones. Robust boundaries were created in order to have boundaries respecting as often as possible the conditions imposed on boundaries in Kozak's algorithm: strata containing at least `minNh` units and positive `nh`. Robust boundaries give an empty take-none stratum if such a stratum is requested, take-all strata as small as possible, and take-some strata with approximately the same number of unique X-values. When `trymany=TRUE`, the same values for the other algorithm parameters (the one given by the user or otherwise the default) are used for every set of initial boundaries. If `trymany=FALSE`, only the given or default (see above) initial boundaries are used. If a user give an `initbh` argument, but also select `trymany=TRUE`, the later is ignored and only the `initbh` initial boundaries are used.

CONVERGENCE

It is possible for the algorithm not to converge. In this case, a warning is printed. The only possible cause of non-convergence for Kozak's algorithm is to reach the maximum number of iterations before the stopping rule has been met. For the algorithm to converge, the argument `maxiter` has to be increased. On the other hand, non-convergence of Sethi's algorithm has several possible causes:

- a division by zero caused by an empty stratum can occur ;
- a division by zero caused by a 0 stratum variance can occur ;
- the square root of a negative number (negative discriminant) can occur ;
- the maximum number of iterations can be reached (often because the algorithm is caught in a loop of non-optimal sets of boundaries).

If a non-convergence happens, the user can try to change the initial boundaries or the model parameters. The user can also choose to work with Kozak's algorithm which should converge given an appropriate maximum number of iterations.

WARNING ABOUT LOCAL MINIMA

Let's note that even if the algorithm converges, it is not guaranteed that it has reached a global minimum. Local minimum can occur both with Sethi's and Kozak's algorithms (Rivest and Baillargeon, 2009).

Value

`bh`	A vector of the $L-1$ optimal stratum boundaries found by the algorithm.
`Nh`	A vector of length L containing the population sizes N_h, i.e. the number of units in each stratum.
`nh`	A vector of length L containing the sample sizes n_h, i.e. the number of units to sample in each stratum. See `stratification-package` for information about the rounding used to get these integer values.
`n`	The total sample size (`sum(nh)`).
`nhnonint`	A vector of length L containing the non-integer values of the sample sizes, obtained directly from applying the allocation rule (see `stratification-package`).
`certain.info`	A vector giving statistics for the certainty stratum (see `stratification-package`). It contains `Nc`, the number of units chosen a priori to be in the sample, and `meanc`, the anticipated mean of Y for these units.
`opti.criteria`	The final value of the criteria to optimize : either n if a target `CV` was given or the RRMSE if a target `n` was given. If the algorithm parameter `idopti` takes the value `"nh"` (the default), the stratum sample sizes used for the calculation of this criteria are the integer ones (`nh`). If the algorithm parameter `idopti` takes the value `"nhnonint"`, the non-integer stratum sample sizes (`nhnonint`) are used. In other words, `idopti` determines here if `opti.criteria` is `opti.nh` or `opti.nhnonint`.
`meanh`	A vector of length L containing the anticipated means of Y in each stratum.
`varh`	A vector of length L containing the anticipated variances of Y in each stratum.

`mean`	A numeric: the anticipated global mean value of Y.
`RMSE`	A numeric: the root mean squared error (or standard error if `takenone`=0) of the anticipated global mean of Y. This is defined as the squared root of: (`bias.penalty` x bias of the mean)^2 + variance of the mean.
`RRMSE`	A numeric: the anticipated relative root mean squared error (or coefficient of variation if `takenone`=0) for the mean of Y, i.e. `RMSE` divided by `mean`.
`relativebias`	A numeric: the anticipated relative bias of the estimator, i.e. (`bias.penalty` x bias of the mean) divided by `mean`. If `takenone`=0, this numeric is zero.
`propbiasMSE`	A numeric: the proportion of the MSE attributable to the bias of the estimator, i.e. (`bias.penalty` x bias of the mean)^2 divided by the MSE of the `mean`. If `takenone`=0, this numeric is zero.
`stratumID`	A factor, having the same length as the input `x`, which values are either 1, 2, ..., L or `"certain"`. The value `"certain"` is given to units a priori chosen to be in the sample. This factor identifies, for each observation, the stratum to which it has been assigned.
`takeall`	The number of take-all strata in the final solution. Note: It is possible that $n_h = N_h$ for non take-all strata because the condition for an automatic addition of a take-all stratum is $n_h > N_h$.
`call`	The function call (object of class "call").
`date`	A character string that contains the system date and time when the function ended.
`args`	A list of all the argument values input to the function or set by default.
`iter.detail`	Only if an iterative algorithm is run (thus not int the output when a complete enumeration is performed): a data frame giving, for each iteration of the algorithm: `"bh"`: the stratum boundaries; `"opti.nh"`: the value of the criteria to optimize calculated with integer stratum sample sizes (for Kozak's algorithm only); `"opti.nhnonint"`: the value of the criteria to optimize calculated with non-integer stratum sample sizes; `"takeall"`: the number of take-all strata for the corresponding boundaries, after making, if needed, an adjustment for non requested take-all strata; `"step"`: the step in the boundary modification (for Kozak's algorithm only); `"iter"`: the iteration identification number; `"run"`: the run identification number (for Kozak's algorithm only) because when `trymany=TRUE` or `rep>1` the algorithm is run more than once. For Kozak's algorithm, a row is added to `iter.detail` only for accepted boundaries, i.e. boundaries decreasing the optimization criteria while respecting the conditions on N_h and n_h (see **Details** for the algorithm description).
`niter`	Only if an iterative algorithm is run: the total number of iterations before convergence of the algorithm. For Kozak's algorithm, this number includes all the iterations, even the ones with discarded boundaries. `niter` is a vector if the algorithm was run more than once and if more than one run lead to the output solution.
`converge`	Only if an iterative algorithm is run: a logical indicating if the algorithm has converged (see **Details**).

`run.detail`	Only for Kozak's algorithm when it is run more than once: a data frame giving the final solution for every run of the algorithm. It contains, for each run: `"bh"`: the stratum boundaries at convergence; `"opti.nh"`: the value of the criteria to optimize calculated with integer stratum sample sizes; `"opti.nhnonint"`: the value of the criteria to optimize calculated with non-integer stratum sample sizes; `"takeall"`: the number of take-all strata for the corresponding boundaries, after making, if needed, an adjustment for non requested take-all strata; `"niter"`: the number of iterations of the algorithm; `"initbh.type"`: the type of initial boundaries requested (`"initbh"` for user-given boundaries, `"cumrootf"`, `"geo"` or `robust`); `"ibh"`: the initial boundaries used to run the algorithm; `"rep"`: the repetition number (because the algorithm is run `rep` times for each set of initial boundaries tried). When `trymany=TRUE`, if a set of initial boundaries do not meet the conditions on N_h and n_h (see **Details** for the algorithm description), the algorithm is not run with these boundaries and no rows are added to `run.detail`.
`run.min`	Only for Kozak's algorithm when it is run more than once: the identification number of every algorithm run leading to the optimal plan, i.e. the rows of `run.detail` containing the proposed solution.
`sol.detail`	Only for Kozak's algorithm when the number of possible sets of boundaries is lower than `minsol`: a data frame giving information for the possible solutions fulfilling the conditions $N_h \geq$ `minNh` and `nh` > 0. It contains: `"bh"`: the stratum boundaries; `"Nh"`: the number of units in each stratum; `"opti.nh"`: the value of the criteria to optimize calculated with integer stratum sample sizes; `"opti.nhnonint"`: the value of the criteria to optimize calculated with non-integer stratum sample sizes; `"takeall"`: the number of take-all strata for the corresponding boundaries, after making, if needed, an adjustment for non requested take-all strata.
`sol.min`	Only for Kozak's algorithm when the number of possible sets of boundaries is lower than `minsol`: the rows of `sol.detail` containing the optimal solution.
`nsol`	Only for Kozak's algorithm : the number of possible sets of boundaries (see **Details**).

Author(s)

Sophie Baillargeon <`Sophie.Baillargeon@mat.ulaval.ca`> and
Louis-Paul Rivest <`Louis-Paul.Rivest@mat.ulaval.ca`>

References

Baillargeon, S., Rivest, L.-P., Ferland, M. (2007). Stratification en enquetes entreprises : Une revue et quelques avancees. *Proceedings of the Survey Methods Section, 2007 SSC Annual Meeting*.

Baillargeon, S. and Rivest, L.-P. (2009). A general algorithm for univariate stratification. *International Stratification Review*, **77**(3), 331-344.

Baillargeon, S. and Rivest L.-P. (2011). The construction of stratified designs in R with the package stratification. *Survey Methodology*, **37**(1), 53-65.

Dalenius, T. and Hodges, J.L., Jr. (1959). Minimum variance stratification. *Journal of the American Statistical Association*, **54**, 88-101.

Gunning, P. and Horgan, J.M. (2004). A new algorithm for the construction of stratum boundaries in skewed populations. *Survey Methodology*, **30**(2), 159-166.

Gunning, P. and Horgan, J.M. (2007). Improving the Lavallee and Hidiroglou algorithm for stratification of skewed populations. *Journal of Statistical Computation and Simulation*, **77**(4), 277-291.

Kozak, M. (2004). Optimal stratification using random search method in agricultural surveys. *Statistics in Transition*, **6**(5), 797-806.

Lavallee, P. and Hidiroglou, M.A. (1988). On the stratification of skewed populations. *Survey Methodology*, **14**, 33-43.

Rivest, L.-P. (1999). Stratum jumpers: Can we avoid them? *Proceedings of the Section on Survey Methods Research of the American Statistical Association*, 64-72.

Rivest, L.-P. (2002). A generalization of the Lavallee and Hidiroglou algorithm for stratification in business surveys. *Survey Methodology*, **28**(2), 191-198.

Sethi, V. K. (1963). A note on optimum stratification of populations for estimating the population means. *The Australian Journal of Statistics*, **5**, 20-33.

See Also

`print.strata`, `plot.strata`, `strata.cumrootf`, `strata.geo`

Examples

```
##########################################################
### Sethi's algorithm versus Kozak's algorithm

# LACK OF CONVERGENCE
# Here is an example of numerical difficulties met with Sethi but not with Kozak
Sethi <- strata.LH(x=UScities, CV=0.01, Ls=3, alloc=c(0.35,0.35,0), takenone=0, takeall=1,
         rh=1, model="loglinear", model.control=list(beta=1, sig2=0.5, ph=0.85),
         algo="Sethi", algo.control=list(maxiter=20))
Sethi
Sethi$iter.detail[1:5,]
# Kozak's algorithm with arithmetic initial boundaries
# (default initial boundaries for Sethi's algorithm)
Kozak<-strata.LH(x=UScities, initbh=c(18,27), CV=0.01, Ls=3, alloc=c(0.35,0.35,0),
       takenone=0, takeall=1, rh=1, model="loglinear",
       model.control=list(beta=1, sig2=0.5, ph=0.85), algo="Kozak")
Kozak
Kozak$iter.detail[Kozak$iter.detail[,"run"]==Kozak$run.min[1],]
# Looking at the iteration history for the optimization with Sethi and Kozak,
# we see that the initial boundaries are very close from the optimal ones.
# Kozak reaches very quickly a minimum. However, Sethi increases n instead of
# minimizing it and afterwards it oscillates between two sets of boundaries
# without converging.
# LOCAL MINIMUM
# In this example, Sethi's algorithm obviously reaches a local minimum since Kozak
# proposes a much smaller n.
Sethi<-strata.LH(x=UScities, CV=0.01, Ls=4, alloc=c(0.5,0,0), takenone=0, takeall=1,
       rh=0.85, model="loglinear", model.control=list(beta=1.1, sig2=0, ph=1),
       algo="Sethi")
Sethi
Kozak<-strata.LH(x=UScities, CV=0.01, Ls=4, alloc=c(0.5,0,0), takenone=0, takeall=1,
       rh=0.85, model="loglinear", model.control=list(beta=1.1, sig2=0, ph=1),
       algo="Kozak")
Kozak


#########################################################
### Take-none stratum

# As illustrated in the following example (presented in Baillargeon and Rivest 2011),
# it is sometimes beneficial to include a take-none stratum in the stratified design
# (possibly with a bias penalty lower than 1).
notn <- strata.LH(x=MRTS, CV=0.1, Ls=3, alloc=c(0.5,0,0.5))
notn
tn1 <- strata.LH(x=MRTS, CV=0.1, Ls=3, alloc=c(0.5,0,0.5), takenone=1)
tn1
tn0.5 <- strata.LH(x=MRTS, CV=0.1, Ls=3, alloc=c(0.5,0,0.5), takenone=1, bias.penalty=0.5)
tn0.5

# Note: Sethi does not converge here. This occurs often with a take-none stratum.
tn1.Sethi <- strata.LH(x=MRTS, CV=0.1, Ls=3, alloc=c(0.5,0,0.5), takenone=1, algo="Sethi")
tn1.Sethi


#########################################################
### Automatic detection of a take-all stratum

# # As in the following example, a beneficial take-all stratum is not always detected
# # by the algorithm. Therefore, it is often a good idea to obtain a stratified design
# # with and without a take-all stratum and to compare the results.
# sans<-strata.LH(x=UScities, n=300, Ls=3, alloc=c(0.35,0.35,0), takeall=0,
#       model="loglinear", model.control=list(beta=0.9, sig2=0, ph=1),
#       algo.control=list(trymany=FALSE))
# sans
# avec<-strata.LH(x=UScities, n=300, Ls=3, alloc=c(0.35,0.35,0), takeall=1,
#       model="loglinear", model.control=list(beta=0.9,sig2=0,ph=1),
#       algo.control=list(trymany=FALSE))
# avec
# # We see that for the target sample size, the anticipated CV is 17% lower with a
# # take-all stratum (0.01081053 vs 0.009002313).


#########################################################
### Models for the discrepancy between Y and X

# LOGLINEAR MODEL WITH MORTALITY: see help(Sweden)
# HETEROSCEDASTIC LINEAR MODEL:  We fix gamma=2.
beta.lin <- mean(Sweden$RMT85/Sweden$REV84)
sig2.lin <- var(Sweden$RMT85/Sweden$REV84)
strata.LH(x=Sweden$REV84, CV=0.05, Ls=5, alloc=c(0.5,0,0.5), takeall=1,
          model="linear", model.control=list(beta=beta.lin, sig2=sig2.lin, gamma=2),
          algo="Kozak")
# Verification of equation 3.6 of Rivest (2002)
beta.log <- 1
sig2.log <- log(1+sig2.lin/beta.lin^2)
strata.LH(x=Sweden$REV84, CV=0.05, Ls=5, alloc=c(0.5,0,0.5), takeall=1,
          model="loglinear", model.control=list(beta=beta.log, sig2=sig2.log, ph=1),
          algo="Kozak")
# The two models give the same stratified design.

# RANDOM REPLACEMENT MODEL: example in Rivest (1999)
strata.LH(x=Sweden$REV84, CV=0.05, Ls=5, alloc=c(0.5,0,0.5), takeall=1,
       model="none", algo="Sethi")     # Table 1 with a different rounding of the nh's
e0 <- strata.LH(x=Sweden$REV84, CV=0.05, Ls=5, alloc=c(0.5,0,0.5), takeall=1,
       model="none", algo="Kozak")
e0                                    # Better than Sethi
var.strata(e0, y=Sweden$RMT85)
e0.001 <- strata.LH(x=Sweden$REV84, CV=0.05, Ls=5, alloc=c(0.5,0,0.5), takeall=1,
          model="random", model.control=list(epsilon=0.011), algo="Kozak")
e0.001                          # Table 2 part 3 with a different rounding of the nh's
var.strata(e0.001 ,y=Sweden$RMT85)
```

`strata.rule` *Non-Iterative Methods of Strata Construction*

Description

These functions first determine boundaries to stratify a population. Then, in a second independent step, the sample sizes are calculated given a CV or the CV is computed given the total sample size. The function `strata.cumrootf` uses the cumulative root frequency method by Dalenius and Hodges (1959) and `strata.geo` uses the geometric method by Gunning and Horgan (2004). A model can be specified for the relationship between the stratification variable X and the survey variable Y, but this model has no impact on the first step of boundary determination. It only influences the calculation of the n or of the CV by the use of anticipated means and variances of Y instead of the empirical means and variances of X.

Usage

```
strata.cumrootf(x, n = NULL, CV = NULL, Ls = 3, certain = NULL,
       alloc = list(q1 = 0.5, q2 = 0, q3 = 0.5), rh = rep(1, Ls),
       model = c("none", "loglinear", "linear", "random"),
       model.control = list(), nclass = NULL)
```

```
strata.geo(x, n = NULL, CV = NULL, Ls = 3, certain=NULL,
       alloc = list(q1 = 0.5, q2 = 0, q3 = 0.5), rh = rep(1, Ls),
       model = c("none", "loglinear", "linear", "random"),
       model.control = list())
```

Arguments

`x`	A vector containing the values of the stratification variable X for every unit in the population.
`n`	A numeric: the target sample size. It has no default value. The argument `n` or the argument `CV` must be input.
`CV`	A numeric: the target coefficient of variation. It has no default value. The argument `CV` or the argument `n` must be input.
`Ls`	A numeric: the number of sampled strata (take-none and certain strata are not counted in `Ls`, but here no take-none stratum can be added to the stratified design so `Ls` is in fact always equal to L). The default is 3.
`certain`	A vector giving the position, in the vector `x`, of the units that must be included in the sample (see `stratification-package`). By default `certain` is `NULL`, which means that no units are chosen a priori to be in the sample.
`alloc`	A list specifying the allocation scheme. The list must contain 3 numerics for the 3 exponents `q1`, `q2` and `q3` in the general allocation scheme (see `stratification-package`). The default is Neyman allocation (`q1=q3=0.5` and `q2=0`)
`rh`	A vector giving the anticipated response rates in each of the `Ls` sampled strata. A single number can be given if the rates do not vary among strata. The default is 1 in each stratum.
`model`	A character string identifying the model used to describe the discrepancy between the stratification variable X and the survey variable Y. It can be `"none"` if one assumes $Y = X$, `"loglinear"` for the loglinear model with mortality, `"linear"` for the heteroscedastic linear model or `"random"` for the random replacement model (see `stratification-package` for a description of these models). The default is `"none"`.
`model.control`	A list of model parameters (see `stratification-package`). The default values of the parameters correspond to the model $Y = X$.
`nclass`	A numeric for the cumulative root frequency method only: the number of classes (Dalenius and Hodges 1959). The default (see **Details**) is `min(Ls*15, Nu)` where `Nu` is the number of unique values in the `x`-vector from which units in the certainty stratum, if any, heve been removed.

Details

The efficiency of the cumulative root frequency method depends on the number of classes `nclass` (see Dalenius and Hodges (1959) for a description of these classes). However, there is no theory about how to choose the best value for `nclass` (Hedlin 2000). This is a limit of the method.

Value

`bh`	A vector of the $L-1$ stratum boundaries proposed by the method.
`nclassh`	A vector for the cumulative root frequency method only: the number of classes in each stratum (Dalenius and Hodges 1959).
`Nh`	A vector of length L containing the population sizes N_h, i.e. the number of units in each stratum.
`nh`	A vector of length L containing the sample sizes n_h, i.e. the number of units to sample in each stratum. See `stratification-package` for information about the rounding used to get these integer values.
`n`	The total sample size (`sum(nh)`).
`nhnonint`	A vector of length L containing the non-integer values of the sample sizes, obtained directly from applying the allocation rule (see `stratification-package`).
`certain.info`	A vector giving statistics for the certainty stratum (see `stratification-package`). It contains `Nc`, the number of units chosen a priori to be in the sample, and `meanc`, the anticipated mean of Y for these units.
`opti.nh`	The final value of the criteria to optimize (either the total sample size n if a target `CV` was given or the RRMSE if a target `n` was given) calculated with the integer stratum sample sizes `nh`.
`opti.nhnonint`	The final value of the criteria to optimize (either the total sample size n if a target `CV` was given or the RRMSE if a target `n` was given) calculated with the non-integer stratum sample sizes `nhnonint`.
`meanh`	A vector of length L containing the anticipated means of Y in each stratum.
`varh`	A vector of length L containing the anticipated variances of Y in each stratum.
`mean`	A numeric: the anticipated global mean value of Y.
`stderr`	A numeric: the standard error of the anticipated global mean of Y.
`CV`	The anticipated coefficient of variation for the mean of Y, i.e. `stderr` divided `mean`.
`stratumID`	A factor, having the same length as the input `x`, which values are either 1, 2, ..., L or `"certain"`. The value `"certain"` is given to units a priori chosen to be in the sample. This factor identifies, for each observation, the stratum to which it has been assigned.
`takeall`	The number of take-all strata in the final solution. Note: It is possible that $n_h = N_h$ for non take-all strata because the condition for an automatic addition of a take-all stratum is $n_h > N_h$.
`call`	The function call (object of class "call").
`date`	A character string that contains the system date and time when the function ended.
`args`	A list of all the argument values input to the function or set by default.

Author(s)

Sophie Baillargeon <`Sophie.Baillargeon@mat.ulaval.ca`> and
Louis-Paul Rivest <`Louis-Paul.Rivest@mat.ulaval.ca`>

References

Baillargeon, S. and Rivest L.-P. (2011). The construction of stratified designs in R with the package stratification. *Survey Methodology*, **37**(1), 53-65.

Dalenius, T. and Hodges, J.L., Jr. (1959). Minimum variance stratification. *Journal of the American Statistical Association*, **54**, 88-101.

Gunning, P. and Horgan, J.M. (2004). A new algorithm for the construction of stratum boundaries in skewed populations. *Survey Methodology*, **30**(2), 159-166.

Hedlin, D. (2000). A procedure for stratification by an extended Ekman rule. *Journal of Official Statistics*, **61**, 15-29.

See Also

`print.strata`, `plot.strata`, `strata.LH`

Examples

```
### Example for strata.cumrootf
res <- matrix(NA, nrow=20, ncol=2)
i <- 1
for ( n in seq(100,2000,100)){
    cum <- strata.cumrootf(x=MRTS, CV=0.01, Ls=4, alloc=c(0.5,0,0.5), nclass=n)
    res[i,] <- c(n,cum$n)
    i <- i + 1
}
plot(res, ylab="suggested sample size n", xlab="number of classes", main=expression(
     paste("Example of the effect of nclass on n for the cum",sqrt(f)," method")))

### Example for strata.geo
strata.geo(x=Sweden$REV84, CV=0.05, Ls=5, alloc=c(0.35,0.35,0), model="none")
strata.geo(x=Sweden$REV84, CV=0.05, Ls=5, alloc=c(0.35,0.35,0), model="loglinear",
       model.control=list(beta=1.058355, sig2=0.06593083, ph=1))
strata.geo(x=Sweden$REV84, CV=0.05, Ls=5, alloc=c(0.35,0.35,0), rh=0.85,
       model="loglinear", model.control=list(beta=1.058355, sig2=0.06593083, ph=1))
# When non-response or a model is added, the stratum boundaries do not change,
# only the nh's do.

### Exemple of how a certainty stratum can be usefull with these methods
strata.cumrootf(x=Sweden$REV84, CV=0.05, Ls=4, alloc=c(0.35,0.35,0), model="none",
                nclass=50)
strata.cumrootf(x=sort(Sweden$REV84), CV=0.05, Ls=4, alloc=c(0.35,0.35,0),
                certain=282:284, model="none", nclass=50)
# The certainty stratum is used here to ensure that the three large units in the
# Sweden$REV84 population are in the sample, since no take-all stratum can be forced
# in the stratified design with the cumulative root frequency or geometric method.
# We see that this allows to reduce by more than half the suggested sample size n
# (47 vs 19). This example was presented in Baillargeon and Rivest (2011).
```

strata.tool *Functions to Visualize Stratified Designs*

Description

`print.strata` prints a "strata" object, presenting the stratification information into a table.

`plot.strata` produces a histogram of the stratification variable X, in which the stratification boundaries are drawn. A table with the `Nh` and `nh` values is also added at the top of the plot.

Usage

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

## S3 method for class 'strata'
plot(x, logscale = FALSE, drop = 0, main =
   paste("Graphical Representation of the Stratified Design", xname),
   xlab, ...)
```

Arguments

`x`	An object of class "strata" to print or to plot.
`logscale`	A logical indicating whether the X axis should be represented on the log scale or not. The default is FALSE.
`drop`	A integer indicating how many of the largest values of X should be omitted in the plot. This argument is useful when some large values of X stretch the X range too much.
`main`	A character string giving the title of the plot.
`xlab`	A character string naming the X axis.
`...`	Additional arguments affecting the print or the plot produced.

Note

When the object of class "strata" contains a certainty stratum, `plot.strata` removes from the data the units in this stratum before generating the histogram.

Author(s)

Sophie Baillargeon <`Sophie.Baillargeon@mat.ulaval.ca`> and
Louis-Paul Rivest <`Louis-Paul.Rivest@mat.ulaval.ca`>

References

Baillargeon, S. and Rivest L.-P. (2011). The construction of stratified designs in R with the package stratification. *Survey Methodology*, **37**(1), 53-65.

See Also

strata.bh, strata.cumrootf, strata.geo, strata.LH

Examples

```
cumrootf <- strata.cumrootf(x=MRTS, CV=0.01, Ls=4, alloc=c(0.5,0,0.5), nclass=500)
print(cumrootf)
plot(cumrootf)
plot(cumrootf, drop=5)
plot(cumrootf, logscale=TRUE)
geo <- strata.geo(x=MRTS, CV=0.01, Ls=4, alloc=c(0.5,0,0.5))
print(geo)
plot(geo, logscale=TRUE)
# The geometric method does not perform well because of some small units
LH <- strata.LH(x=MRTS, CV=0.01, Ls=4, alloc=c(0.5,0,0.5), takeall=1)
print(LH)
plot(LH, logscale=TRUE)
```

Sweden *The MU284 Population of Sweden Municipalities from Sarndal et al. (1992)*

Description

This data set comes from Sarndal et al.'s book (1992), Appendix B. It contains different variables that describe 284 municipalities in Sweden.

Usage

```
Sweden
```

Format

A data frame with 284 observations on the following 11 variables.

`id` Identifier running from 1 to 284

`P85` 1985 population (in thousands)

`P75` 1975 population (in thousands)

`RMT85` Revenues from the 1985 municipal taxation (in millions of kronor)

`CS82` Number of Conservative seats in municipal council

`SS82` Number of Social-Democratic seats in municipal council

`S82` Total number of seats in municipal council

`ME84` Number of municipal employees in 1984

`REV84` Real estate values according to 1984 assessment (in millions of kronor)

`REG` Geographic region indicator

`CL` Cluster indicator (a cluster consists of a set of neighboring municipalities)

Details

In this package, REV84 is used as a stratification variable and RMT85 as a survey variable.

Source

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

References

Rivest, L.-P. (2002). A generalization of the Lavallee and Hidiroglou algorithm for stratification in business surveys. *Survey Methodology*, **28**(2), 191-198.

Examples

```
X <- Sweden$REV84
Y <- Sweden$RMT85

# Study of the relationship between X and Y
plot(log(X), log(Y))
# Extreme values are omitted for a more robust estimation
keep <- X/Y>quantile(X/Y,0.03)&X/Y<quantile(X/Y,0.97)
plot(log(X)[keep], log(Y)[keep])
reg<-lm( log(Y)[keep]~log(X)[keep] )
summary(reg)

# Stratification assuming X=Y
nomodel <- strata.LH(x=X, CV=0.05, Ls=3, alloc=c(0.5,0,0.5), takeall=1, model="none")
nomodel
var.strata(nomodel, y=Y) # The target CV is not reached

# Stratification taking into account a loglinear model between X and Y,
# using the estimated parameters values
model <- strata.LH(x=X, CV=0.05, Ls=3, alloc=c(0.5,0,0.5), takeall=1, model="loglinear",
        model.control=list(beta=reg$coef[2], sig2=summary(reg)$sigma^2, ph=1))
model
var.strata(model, y=Y) # The target CV is reached
```

---

var.strata *Anticipated Variances and RRMSE from a Stratified Design for a Survey Variable Y*

Description

`var.strata` calculates the anticipated means, variances and relative root mean squared error (RRMSE) obtained when applying a stratified design to a survey variable Y. The variable Y can be input or it can be defined from X by a specified loglinear with mortality, heteroscedastic linear or random replacement model.

`print.var.strata` prints a "var.strata" object, presenting the stratification information into a table.

Usage

```
var.strata(strata, y = NULL, rh = strata$args$rh, rh.postcorr =
          FALSE, model = c("none", "loglinear", "linear", "random"),
          model.control = list())

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

Arguments

`strata`	An object of class "strata", which represents a stratified design.
`y`	A vector containing the values of the survey variable Y for every unit of the population, respecting the order of the units in the `x`-vector used to create `strata`. The default is that Y is not given.
`rh`	A vector giving the anticipated response rates in each of the `Ls` sampled strata. A single number can be given if the rates do not vary among strata. The default is to use the rates given in the `strata.bh` object.
`rh.postcorr`	A logical. If TRUE, a posterior correction for non-response is applied. This correction takes into account the non-response in the `strata.bh` object. It is only available when the stratified design `strata` had a target CV. The default is FALSE, i.e. no posterior correction is made (see **Details**).
`model`	A character string identifying the model used to describe the discrepancy between the stratification variable X and the survey variable Y. It can be `"none"` if one assumes $Y = X$, `"loglinear"` for the loglinear model with mortality, `"linear"` for the heteroscedastic linear model or `"random"` for the random replacement model (see `stratification-package` for a description of these models). The default is `"none"`.
`model.control`	A list of model parameters (see `stratification-package`). The default values of the parameters correspond to the model $Y = X$.
`x`	An object of class "var.strata" to print.
`...`	Additional arguments affecting the print produced.

Details

POSTERIOR CORRECTION FOR NON-RESPONSE (with a target CV only

The optional posterior correction for non-response is done as follows. For each take-some stratum, n_h is increased if the input `rh` is lower than the anticipated response rate in the `strata.bh` object, and n_h is decreased if the input `rh` is higher than the anticipated response rate given when creating the `strata.bh` object. The modification of n_h is done by multiplying it by `strata$args$rh/rh`.

The weakness of this posterior correction is that it cannot take into account non-response in a take-all stratum. In that stratum, n_h cannot be increased since it is equal to N_h. To correctly account for non-response in a take-all stratum, the boundary of the stratum has to be lowered. This is what the generalized Lavallee-Hidiroglou method does (`strata.LH`).

Value

`nh`	A vector of length `L` containing the integer sample sizes n_h, i.e. the number of units to sample in each stratum. This vector can be different than `strata$nh` if `rh.postcorr=TRUE`.
`n`	The total sample size (`sum(nh)`). This number can be different than `strata$n` if `rh.postcorr=TRUE`.
`nhnonint`	A vector of length L containing the non-integer values of the sample sizes. This vector can be different than `strata$nhnonint` if `rh.postcorr=TRUE`.
`certain.info`	A vector giving statistics for the certainty stratum (see `stratification-package`). It contains `Nc`, the number of units chosen a priori to be in the sample, and `meanc`, the anticipated mean of Y for these units.
`meanh`	A vector of length L containing the anticipated means of Y in each stratum.
`varh`	A vector of length L containing the anticipated variances of Y in each stratum.
`mean`	A numeric: the anticipated global mean value of Y.
`RMSE`	A numeric: the root mean squared error (or standard error if `strata$args$takenone`=0) of the anticipated global mean of Y. This is defined as the squared root of: (`bias.penalty` x bias of the mean)^2 + variance of the mean.
`RRMSE`	A numeric: the anticipated relative root mean squared error (or coefficient of variation if `strata$args$takenone`=0) for the mean of Y, i.e. `RMSE` divided by `mean`.
`relativebias`	A numeric: the anticipated relative bias of the estimator, i.e. (`bias.penalty` x bias of the mean) divided by `mean`. If `strata$args$takenone`=0, this numeric is zero.
`propbiasMSE`	A numeric: the proportion of the MSE attributable to the bias of the estimator, i.e. (`bias.penalty` x bias of the mean)^2 divided by the MSE of the `mean`. If `strata$args$takenone`=0, this numeric is zero.
`call`	The function call (object of class "call").
`date`	A character string that contains the system date and time when the function ended.
`args`	A list of all the arguments input to the function or used by default.

Author(s)

Sophie Baillargeon <`Sophie.Baillargeon@mat.ulaval.ca`> and
Louis-Paul Rivest <`Louis-Paul.Rivest@mat.ulaval.ca`>

References

Baillargeon, S. and Rivest L.-P. (2011). The construction of stratified designs in R with the package stratification. *Survey Methodology*, **37**(1), 53-65.

See Also

`strata.bh`, `strata.cumrootf`, `strata.geo`, `strata.LH`

Examples

```
nomodel <- strata.LH(x=Sweden$REV84, CV=0.05, Ls=3, alloc=c(0.5,0,0.5),
          takeall=1, model="none")
# We can give a vector of the Y values for every unit in the population
var.strata(nomodel, y=Sweden$RMT85)
# Or specify a model between X and Y
var.strata(nomodel, model="loglinear", model.control=list(beta=1.058355,
           sig2=0.06593083, ph=1))
# Compared to taking into account the model in the optimization
model <- strata.LH(x=Sweden$REV84, CV=0.05, Ls=3, alloc=c(0.5,0,0.5),
         takeall=1, model="loglinear", model.control=list(beta=1.058355,
     sig2=0.06593083, ph=1))
var.strata(model, y=Sweden$RMT85)

### Examples of posterior correction for non-response
LH <- strata.LH(x=MRTS, CV=0.01, Ls=4, alloc=c(0.5,0,0.5), takeall=1)
LH
# Without non-response in the take-all strata
var.strata(LH, rh.postcorr=TRUE, rh=c(0.85,0.9,0.9,1))
strata.LH(x=MRTS, CV=0.01, Ls=4, alloc=c(0.5,0,0.5), takeall=1, rh=c(0.85,0.9,0.9,1))
# With non-response in the take-all strata
var.strata(LH, rh.postcorr=TRUE, rh=0.9)
strata.LH(x=MRTS, CV=0.01, Ls=4, alloc=c(0.5,0,0.5), takeall=1, rh=0.9)
```

Index