

Package ‘SamplingStrata’

January 12, 2016

Type Package

Title Optimal Stratification of Sampling Frames for Multipurpose
Sampling Surveys

Version 1.1

Date 2016-01-12

Author Giulio Barcaroli, Daniela Pagliuca, Egon Willighagen,
Diego Zardetto

Maintainer Giulio Barcaroli <barcarol@istat.it>

Description In the field of stratified sampling design, this package offers an approach for the determination of the best stratification of a sampling frame, the one that ensures the minimum sample cost under the condition to satisfy precision constraints in a multivariate and multidomain case. This approach is based on the use of the genetic algorithm: each solution (i.e. a particular partition in strata of the sampling frame) is considered as an individual in a population; the fitness of all individuals is evaluated applying the Bethel-Chromy algorithm to calculate the sampling size satisfying precision constraints on the target estimates. Functions in the package allows to: (a) analyse the obtained results of the optimisation step; (b) assign the new strata labels to the sampling frame; (c) select a sample from the new frame accordingly to the best allocation. Functions for the execution of the genetic algorithm are a modified version of the functions in the 'genalg' package.

License GPL (>= 2)

LazyLoad yes

Depends R (>= 2.15.0), memoise

R topics documented:

bethel	2
buildStrataDF	3
checkInput	4
errors	5
evalSolution	6
optimizeStrata	7

plotSamprate	10
selectSample	11
strata	12
swisserrors	13
swissframe	14
swissmunicipalities	15
swissstrata	16
tuneParameters	17
updateFrame	19
updateStrata	21
var.bin	22

Index	24
--------------	-----------

bethel	<i>Multivariate optimal allocation</i>
--------	--

Description

Multivariate optimal allocation for different domains of interest in stratified sample design

Usage

```
bethel (
  stratif,
  errors,
  minnumstrat=2,
  maxiter=200,
  maxiter1=25,
  printa=FALSE,
  realAllocation=FALSE,
  epsilon=1e-11
)
```

Arguments

errors	Data frame of coefficients of variation for each domain
stratif	Data frame of survey strata
minnumstrat	Minimum number of units per strata (default=2)
maxiter	Maximum number of iterations of the algorithm (default=200)
maxiter1	Maximum number of iterations (default=25) of the general procedure. This kind of iteration may be required by the fact that when in a stratum the number of allocated units is greater or equal to its population, that stratum is set as "census stratum", and the whole procedure is re-initialised
printa	If TRUE then two attributes are added to the resulting vector. The first ('confr') is a comparison between results obtained with 3 different allocation methods: Bethel, proportional and equal. The second ('outcv') is a table reporting planned and actual CV, together with a sensitivity analysis
realAllocation	If FALSE, the allocation is based on INTEGER values; if TRUE, the allocation is based on REAL values

epsilon Epsilon (default=1e-11): this value is used to compare the difference in results from one iteration to the other; if it is lower than "epsilon", then the procedure stops

Value

A vector containing the computed optimal allocation

Author(s)

Daniela Pagliuca with contributions from Teresa Buglielli and Giulio Barcaroli

Examples

```
library(SamplingStrata)
data(strata)
data(errors)
n <- bethel(strata, errors, printa=TRUE)
sum(n)
```

buildStrataDF	<i>Builds the "strata" dataframe containing information on target variables Y's distributions in the different strata, starting from sample data or from a frame</i>
---------------	--

Description

This function allows to build the information regarding strata in the population required as an input by the algorithm of Bethel for the optimal allocation. In order to estimate means and standard deviations for target variables Y's, we need data coming from: (1) a previous round of the survey whose sample we want to plan; (2) sample data from a survey with variables that are proxy to the ones we are interested to; (3) a frame containing values of Y's variables (or proxy variables) for all the population. In all cases, each unit in the dataset must contain auxiliary information (X's variables) and also target variables Y's (or proxy variables) values: under these conditions it is possible to build the dataframe "strata", containing information on the distribution of Y's in the different strata (namely, means and standard deviations), together with information on strata (total population, if it is to be censused or not, the cost per single interview). If the information is contained in a sample dataset, a variable named WEIGHT is expected to be present. In case of a frame, no such variable is given, and the function will define a WEIGHT variable for each unit, whose value is always '1'. Missing values for each Y variable will not be taken into account in the computation of means and standard deviations (in any case, NA's can be present in the dataset). The dataframe "strata" is written to an external file (tab delimited, extension "txt"), and will be used as an input by optimizeStrata.

Usage

```
buildStrataDF(dataset)
```

Arguments

`dataset` This is the name of the dataframe containing the sampling data, or frame data. It is strictly required that auxiliary information is organised in variables named as X_1, X_2, \dots, X_m (there should be at least one of them) and the target variables are denoted by Y_1, Y_2, \dots, Y_n . In addition, in case of sample data, a variable named 'WEIGHT' must be present in the dataframe, containing the weights associated to each sampling unit

Value

A dataframe containing strata

Author(s)

Giulio Barcaroli

Examples

```
data(swissframe)
strata <- buildStrataDF(swissframe)
head(strata)
```

checkInput	<i>Checks the inputs to the package: dataframes "errors", "strata" and "sampling frame"</i>
------------	---

Description

This functions checks the internal structure of the different input dataframes ("errors", "strata" and "sampling frame"), and also the correctness of the relationships among them.

Usage

```
checkInput(errors=NULL, strata=NULL, sampframe=NULL)
```

Arguments

`errors` Dataframe containing the precision levels expressed in terms of maximum acceptable coefficients of variation that estimates of target variables Y 's of the survey must comply.

`strata` Dataframe containing the information related to strata.

`sampframe` Dataframe containing the information related to all the units belonging to the population of interest.

Author(s)

Giulio Barcaroli

Examples

```
library(SamplingStrata)
data(swisserrors)
data(swisstrata)
data(swissframe)
checkInput(swisserrors, swisstrata, swissframe)
checkInput(strata=swisstrata, sampframe=swissframe)
checkInput(strata=swisstrata)
```

errors

Precision constraints (maximum CVs) as input for Bethel allocation

Description

Dataframe containing precision levels (expressed in terms of acceptable CV's)

Usage

```
data(errors)
```

Format

The constraint data frame (errors) contains a row per each domain value with the following variables:

DOM Type of domain code (factor)

CV1 Planned coefficient of variation for first variable Y1 (numeric)

CVj Planned coefficient of variation for j-th variable Yj (numeric)

CVn Planned coefficient of variation for last variable Yn (numeric)

domainvalue Value of the domain to which the constraints refer (numeric)

Details

Note: the names of the variables must be the ones indicated above

Examples

```
## data(errors)
## errors
```

evalSolution	<i>Allows to evaluate the solution produced by the function 'optimizeStrata' by selecting a number of samples from the frame with the optimal stratification, and calculating average CV's on the target variables Y's together with differences between estimates and the values of the parameters in the population.</i>
--------------	--

Description

The user can indicate the number of samples that must be selected by the frame to which the optimal stratification has been applied. The allocation is the one reported in the dataframe 'outstrata'. First, the true values of the parameters are calculated from the frame. Then, for each sample the sampling estimates are calculated, together with the differences between them and the true values of the parameters. At the end, an estimate of the CV is produced for each target variable, in order to compare them with the precision constraints set at the beginning of the optimization process. If the flag 'writeFiles' is set to TRUE, boxplots of distribution of the CV's in the different domains are produced for each Y variable ('cv.pdf'), together with boxplot of the distributions of differences between estimates and values of the parameters in the population ('differences.pdf').

Usage

```
evalSolution(frame,
             outstrata,
             nsampl=100,
             writeFiles=FALSE)
```

Arguments

frame	The frame to which the optimal stratification has been applied ('framenew')
outstrata	The new (aggregated) strata generated by the function 'optimizeStrata'
nsampl	The number of sample to be drawn from the frame
writeFiles	A flag to write in the work directory the outputs of the function

Value

A list containing (1) the vector of the solution and (2) the optimal aggregated strata

Author(s)

Giulio Barcaroli

Examples

```
## Not run:
library(SamplingStrata)
data(swisserrors)
data(swissstrata)
solution <- optimizeStrata (
  errors = swisserrors,
  strata = swissstrata,
  cens = NULL,
```

```

strcens = FALSE,
alldomains = TRUE,
dom = dominio,
initialStrata = nrow(swissstrata)/3,
addStrataFactor = 0.01,
minnumstr = 2,
iter = 50,
pops = 20,
mut_chance = 0.05,
elitism_rate = 0.2,
highvalue = 100000000,
suggestions = NULL,
realAllocation = TRUE,
writeFiles = FALSE)
# update sampling strata with new strata labels
#swissstrata <- swissstrata[swissstrata$DOM1 == dominio,]
newstrata <- updateStrata(swissstrata, solution, writeFiles = TRUE)
# update sampling frame with new strata labels
data(swissframe)
framewnew <- updateFrame(frame=swissframe,newstrata=newstrata,writeFile=TRUE)
samp <- selectSample(framewnew,solution$aggr_strata,writeFiles=TRUE)
# evaluate the current solution
results <- evalSolution(framewnew, solution$aggr_strata, 100, writeFiles = TRUE)
boxplot(val ~ cv, data = results$coeff_var,
        col = "orange",
        main = "Distribution of CVs in the domains",
        xlab = "Variables Y",
        ylab = "Value of CV")
diff <- read.csv("differences.csv")
numY <- sum(grepl("diff", colnames(diff)))
k <- ceiling(numY/4)
for (j in 1:k) {
  split.screen(c(2, 2))
  for (i in 1:4) {
    if (i + 4 * (j - 1) <= numY) {
      stmt <- paste("screen(", i, ")", sep = "")
      eval(parse(text = stmt))
      stmt <- paste("boxplot(diff",i,"~dom,
                        data=diff,ylab=Differences,
                        xlab=Domain,col = orange)",sep="")
      eval(parse(text = stmt))
      stmt <- paste("mtext(expression(Y", i , ")", side=3, adj=0, cex=1.0, line=1)",
                    sep = "")
      eval(parse(text = stmt))
    }
  }
}
## End(Not run)

```

optimizeStrata

Best stratification of a sampling frame for multipurpose surveys

Description

This function runs a set of other functions to optimise the stratification of a sampling frame

Usage

```
optimizeStrata(
  errors ,
  strata ,
  cens = NULL,
  strcens = FALSE,
  alldomains = TRUE,
  dom = NULL,
  initialStrata = nrow(strata),
  addStrataFactor = 0.01,
  minnumstr = 2,
  iter = 20,
  pops = 20,
  mut_chance = 0.05,
  elitism_rate = 0.2,
  highvalue = 1e+08,
  suggestions = NULL,
  realAllocation = FALSE,
  writeFiles = FALSE,
  showPlot = TRUE
)
```

Arguments

errors	This is the (mandatory) dataframe containing the precision levels expressed in terms of Coefficients of Variation that estimates on target variables Y's of the survey must comply
strata	This is the (mandatory) dataframe containing the information related to "atomic" strata, i.e. the strata obtained by the Cartesian product of all auxiliary variables X's. Information concerns the identifiability of strata (values of X's) and variability of Y's (for each Y, mean and standard error in strata)
cens	This the (optional) dataframe containing the takeall strata, those strata whose units must be selected in whatever sample. It has same structure than "strata" dataframe
strcens	Flag (TRUE/FALSE) to indicate if takeall strata do exist or not. Default is FALSE
alldomains	Flag (TRUE/FALSE) to indicate if the optimization must be carried out on all domains (default is TRUE). If it is set to FALSE, then a value must be given to parameter 'dom'
dom	Indicates the domain on which the optimization must be carried. It is an integer value that has to be internal to the interval (1 <-> number of domains). If 'alldomains' is set to TRUE, it is ignored
initialStrata	This is the initial limit on the number of strata for each solution. Default is 3000
addStrataFactor	This parameter indicates the probability that at each mutation the number of strata may increase with respect to the current value. Default is 0.01 (1
minnumstr	Indicates the minimum number of units that must be allocated in each stratum. Default is 2
iter	Indicated the maximum number of iterations (= generations) of the genetic algorithm. Default is 20

pops	The dimension of each generations in terms of individuals. Default is 50
mut_chance	Mutation chance: for each new individual, the probability to change each single chromosome, i.e. one bit of the solution vector. High values of this parameter allow a deeper exploration of the solution space, but a slower convergence, while low values permit a faster convergence, but the final solution can be distant from the optimal one. Default is 0.05
elitism_rate	This parameter indicates the rate of better solutions that must be preserved from one generation to another. Default is 0.2 (20
highvalue	Parameter for genetic algorithm. Not to be changed
suggestions	Optional parameter for genetic algorithm that indicates one possible solution (maybe from previous runs) that will be introduced in the initial population. Default is NULL.
realAllocation	If FALSE, the allocation is based on INTEGER values; if TRUE, the allocation is based on REAL values
writeFiles	Indicates if the various dataframes and plots produced during the execution have to be written in the working directory. Default is "FALSE".
showPlot	Indicates if the plot showing the trend in the value of the objective function has to be shown or not. Default is "TRUE".

Value

A list containing (1) the vector of the solution and (2) the optimal aggregated strata

Author(s)

Giulio Barcaroli

Examples

```
#
# This is a toy example, and can be run
#

library(SamplingStrata)
data(errors)
data(strata)
# optimisation of sampling strata
solution <- optimizeStrata (
  errors = errors,
  strata = strata,
  cens = NULL,
  strcens = FALSE,
  initialStrata = 3000,
  addStrataFactor = 0.01,
  minnumstr = 2,
  iter = 30,
  pops = 20,
  mut_chance = 0.05,
  elitism_rate = 0.2,
  highvalue = 100000000,
  suggestions = NULL,
writeFile = FALSE,
showPlot = TRUE)
```

```

sum(ceiling(solution$aggr_strata$SOLUZ))
head(solution$aggr_strata)

#
# The following example is realistic, but is time consuming
#
## Not run:
library(SamplingStrata)
data(swisserrors)
data(swissstrata)
# optimisation of sampling strata
solution <- optimizeStrata (
  errors = swisserrors,
  strata = swissstrata,
  cens = NULL,
  strcens = FALSE,
  initialStrata = 3000,
  addStrataFactor = 0.01,
  minnumstr = 2,
  iter = 60,
  pops = 20,
  mut_chance = 0.05,
  elitism_rate = 0.2,
  highvalue = 100000000,
  suggestions = NULL,
writeFile = FALSE,
showPlot = TRUE)
sum(ceiling(solution$aggr_strata$SOLUZ))
head(solution$aggr_strata)

## End(Not run)

```

plotSamprate

Plotting sampling rates in the different strata for each domain in the solution.

Description

Once the optimization step has been carried out, by applying this function it is possible to obtain the visualization of the proportion of sampling units in the different strata for each domain in the obtained solution.

Usage

```
plotSamprate(solution, dom)
```

Arguments

solution	Solution obtained by executing optimizeStrata
dom	Identification of the domain

Value

Plot

Examples

```
## Not run:
library(SamplingStrata)
data(swisserrors)
data(swissstrata)
# optimisation of sampling strata
solution <- optimizeStrata (
  errors = swisserrors,
  strata = swissstrata,
  cens = NULL,
  strcens = FALSE,
  initialStrata = 3000,
  addStrataFactor = 0.01,
  minnumstr = 2,
  iter = 60,
  pops = 20,
  mut_chance = 0.05,
  elitism_rate = 0.2,
  highvalue = 100000000,
  suggestions = NULL,
writeFiles = FALSE)
# plot of the sampling rates in strata of domain 1
plotSamprate(solution, 1)

## End(Not run)
```

selectSample

*Selection of a stratified sample from the frame with srswor method***Description**

Once optimal stratification has been obtained (in the dataframe 'outstrata'), and a new frame has been built by assigning to the units of the old one the new stratum labels (by means of "updateFrame" function), it is possible to select a stratified sample from the frame with the srswor method. The result of the execution of "selectSample" function is a dataframe containing selected units, with the probabilities of inclusion. It is possible to output this dataframe in a .csv file. One more .csv file is produced ("sampling check"), containing coherence checks between (a) population in frame strata (b) population in optimised strata (c) planned units to be selected in optimised strata (d) actually selected units (e) sum of weights in each stratum

Usage

```
selectSample(frame, ostrata, writeFiles = FALSE,verbatim=TRUE)
```

Arguments

frame	This is the (mandatory) dataframe containing the sampling frame, as it has been modified by the execution of the "updateFrame" function. Name of stratum variable must be 'strato'.
ostrata	This is the (mandatory) dataframe containing the information related to resulting stratification obtained by the execution of "optimizeStrata" function. Name of stratum variable must be 'strato'.

writeFiles	Indicates if at the end of the processing the resulting strata will be outputted in a delimited file. Default is "FALSE".
verbatim	Indicates if information on the drawn sample must be printed or not. Default is "TRUE".

Value

A dataframe containing the sample

Author(s)

Giulio Barcaroli with contribution from Diego Zardetto

Examples

```
#
# The following example is realistic, but is time consuming
#
## Not run:
library(SamplingStrata)
data(swisserrors)
data(swissstrata)
# optimisation of sampling strata
solution <- optimizeStrata (
  errors = swisserrors,
  strata = swissstrata,
  cens = NULL,
  strcens = FALSE,
  initialStrata = 3000,
  addStrataFactor = 0.01,
  minnumstr = 2,
  iter = 60,
  pops = 20,
  mut_chance = 0.05,
  elitism_rate = 0.2,
  highvalue = 100000000,
  suggestions = NULL,
writeFiles = FALSE)
# updating sampling strata with new strata labels
newstrata <- updateStrata(swissstrata, solution)
# updating sampling frame with new strata labels
data(swissframe)
framewnew <- updateFrame(frame=swissframe,newstrata=newstrata)
# selection of sample
sample <- selectSample(frame=framewnew,outstrata=solution$aggr_strata)
head(sample)

## End(Not run)
```

strata

Dataframe containing information on strata in the frame

Description

Dataframe containing information on strata in the frame

Usage

```
data(strata)
```

Format

The strata data frame (strata) contains a row per stratum with the following variables:

- stratum** Identifier of the stratum (numeric)
- N** Number of population units in the stratum (numeric)
- X1** Value of first auxiliary variable X1 in the stratum (factor)
- Xi** Value of i-th auxiliary variable Xi in the stratum (factor)
- Xk** Value of last auxiliary variable Xk in the stratum (factor)
- M1** Mean in the stratum of the first variable Y1 (numeric)
- Mj** Mean in the stratum of the j-th variable Yt (numeric)
- Mn** Mean in the stratum of the last variable Y (numeric)
- S1** Standard deviation in the stratum of the first variable Y (numeric)
- Sj** Standard deviation in the stratum of the j-th variable Yt (numeric)
- Sn** Standard deviation in the stratum of the last variable Y (numeric)
- cons** Flag (1 indicates a take all stratum, 0 a sampling stratum) (numeric) Default = 0
- cost** Cost per interview in each stratum. Default = 1 (numeric)
- DOM1** Value of domain to which the stratum belongs (factor or numeric)

Details

Note: the names of the variables must be the ones indicated above

Examples

```
# data(strata)
# head(strata)
```

 swisserrors

Precision constraints (maximum CVs) as input for Bethel allocation

Description

Dataframe containing precision levels (expressed in terms of acceptable CV's)

Usage

```
data(errors)
```

Format

The constraint data frame (swisserrors) contains a row per each domain value with the following variables:

DOM Type of domain code (factor)

CV1 Planned coefficient of variation for first variable Y1 (number of men and women aged between 0 and 19) (numeric)

CV2 Planned coefficient of variation for second variable Y2 (number of men and women aged between 20 and 39) (numeric)

CV3 Planned coefficient of variation for third variable Y3 (number of men and women aged between 40 and 64) (numeric)

CV4 Planned coefficient of variation for fourth variable Y4 (number of men and women aged between 65 and over) (numeric)

domainvalue Value of the domain to which the constraints refer (numeric)

Examples

```
## data(swisserrors)
## swisserrors
```

swissframe

Dataframe containing information on all units in the population of reference that can be considered as the final sampling unit (this example is related to Swiss municipalities)

Description

Dataframe containing information on all municipalities in Swiss (it is a derivation of dataframe "swissmunicipalities" in "sampling" package)

Usage

```
data(swissframe)
```

Format

The "swissframe" dataframe contains a row per each Swiss municipality with the following variables:

progr Progressive associated to the frame unit (numeric)

id Name of the frame unit (character)

X1 Classes of total population in the municipality (factor with 18 values)

X2 Classes of wood area in the municipality (factor with 3 values)

X3 Classes of area under cultivation in the municipality (factor with 3 values)

X4 Classes of mountain pasture area in the municipality (factor with 3 values)

X5 Classes of area with buildings in the municipality (factor with 3 values)

X6 Classes of industrial area in the municipality (factor with 3 values)

Y1 Number of men and women aged between 0 and 19 (numeric)

Y2 Number of men and women aged between 20 and 39 (numeric)

Y3 Number of men and women aged between 40 and 64 (numeric)

Y4 Number of men and women aged between 65 and over (numeric)

domainvalue Value of domain to which the municipality belongs (factor or numeric)

Examples

```
#data(swissframe)
#head(strata)
```

swissmunicipalities *The Swiss municipalities population*

Description

This population provides information about the Swiss municipalities in 2003.

Usage

```
data(swissmunicipalities)
```

Format

A data frame with 2896 observations on the following 22 variables:

CT Swiss canton.

REG Swiss region.

COM municipality number.

Nom municipality name.

HApoly municipality area.

Surfacesbois wood area.

Surfacescult area under cultivation.

Alp mountain pasture area.

Airbat area with buildings.

Airind industrial area.

P00BMTOT number of men.

P00BWTOT number of women.

Pop020 number of men and women aged between 0 and 19.

Pop2040 number of men and women aged between 20 and 39.

Pop4065 number of men and women aged between 40 and 64.

Pop65P number of men and women aged between 65 and over.

H00PTOT number of households.

H00P01 number of households with 1 person.

H00P02 number of households with 2 persons.

H00P03 number of households with 3 persons.

H00P04 number of households with 4 persons.

POPTOT total population.

Source

Swiss Federal Statistical Office.

Examples

```
# data(swissmunicipalities)
# hist(swissmunicipalities$POPTOT)
```

swissstrata	<i>Dataframe containing information on strata in the swiss municipalities frame</i>
-------------	---

Description

Dataframe containing information on strata in the swiss municipalities frame

Usage

```
data(swissframe)
```

Format

The "swissstrata" dataframe contains a row per stratum with the following variables:

- STRATO** Identifier of the stratum (character)
- N** Number of population units in the stratum (numeric)
- X1** Classes of total population in the municipality (factor with 18 values)
- X2** Classes of wood area in the municipality (factor with 3 values)
- X3** Classes of area under cultivation in the municipality (factor with 3 values)
- X4** Classes of mountain pasture area in the municipality (factor with 3 values)
- X5** Classes of area with buildings in the municipality (factor with 3 values)
- X6** Classes of industrial area in the municipality (factor with 3 values)
- M1** Mean in the stratum of Y1 (number of men and women aged between 0 and 19)(numeric)
- M2** Mean in the stratum of Y2 (number of men and women aged between 20 and 39) (numeric)
- M3** Mean in the stratum of Y3 (number of men and women aged between 40 and 64) (numeric)
- M4** Mean in the stratum of Y4 (number of men and women aged between 64 and over) (numeric)
- S1** Standard deviation in the stratum of Y1 (number of men and women aged between 0 and 19)(numeric)
- S2** Standard deviation in the stratum of Y2 (number of men and women aged between 20 and 39) (numeric)
- S3** Standard deviation in the stratum of Y3 (number of men and women aged between 40 and 64) (numeric)
- S4** Standard deviation in the stratum of Y4 (number of men and women aged between 64 and over) (numeric)
- cens** Flag (1 indicates a take all stratum, 0 a sampling stratum) (numeric) Default = 0
- cost** Cost per interview in each stratum. Default = 1 (numeric)
- DOM1** Value of domain to which the stratum belongs Default = 1 (factor or numeric)

Examples

```
# data(swisstrata)
# head(swisstrata)
```

tuneParameters

Execution and compared evaluation of optimization runs

Description

This function allows to execute a number of optimization runs, varying in a controlled way the values of the parameters, in order to find their most suitable values. by comparing the resulting solutions. It can be applied only to a given domain per time. Most parameters of this function are the same than those of the function 'optimizeStrata', but they are given in a vectorial format. The length of each vector is given by the number of optimizations to be run: it is therefore possible to define different combination of values of the parameters for each execution of 'optimizeStrata'. After each optimization run, from the corresponding optimized frame a given number of samples are drawn. For each of them, the estimates of the target variables Y's are computed ("precision"), together with the associated coefficients of variations, and the absolute differences between the values of the estimates and the true values in the population ("bias"). Information on the distribution of bias (differences) and precision (CV's) are outputted, and also boxplots for each of them are produced, in order to permit a compared evaluation of the different solutions found in the different runs. As the optimal solution is stored for each run, after the evaluation it is possible to use it directly, or as a "suggestion" for a new optimization with more iterations (in order to improve it).

Usage

```
tuneParameters (
  noptim,
  nsampl,
  frame,
  errors = errors,
  strata = strata,
  cens = NULL,
  strcens = FALSE,
  alldomains = FALSE,
  dom = 1,
  initialStrata,
  addStrataFactor,
  minnumstr,
  iter,
  pops,
  mut_chance,
  elitism_rate
)
```

Arguments

noptim	Number of optimization runs to be performed
nsampl	Number of samples to be drawn from the optimized population frame after each optimization

frame	The (mandatory) dataframe containing the sampling frame
errors	This is the (mandatory) dataframe containing the precision levels expressed in terms of Coefficients of Variation that estimates on target variables Y's of the survey must comply
strata	This is the (mandatory) dataframe containing the information related to "atomic" strata, i.e. the strata obtained by the Cartesian product of all auxiliary variables X's. Information concerns the identifiability of strata (values of X's) and variability of Y's (for each Y, mean and standard error in strata)
cens	This the (optional) dataframe containing the takeall strata, those strata whose units must be selected in whatever sample. It has same structure than "strata" dataframe
strcens	Flag (TRUE/FALSE) to indicate if takeall strata do exist or not. Default is FALSE
alldomains	Flag (TRUE/FALSE) to indicate if the optimization must be carried out on all domains. It must be left to its default (FALSE)
dom	Indicates the domain on which the optimization runs must be performed. It is an integer value that has to be internal to the interval (1 <-> number of domains). It is mandatory, if not indicated, the default (1) is taken.
initialStrata	This is the initial limit on the number of strata for each solution. Default is 3000. This parameter has to be given in a vectorial format, whose length is given by the number of different optimisations (= value of parameter 'noptim')
addStrataFactor	This parameter indicates the probability that at each mutation the number of strata may increase with respect to the current value. Default is 0.01 (1 This parameter has to be given in a vectorial format, whose length is given by the number of different optimisations (= value of parameter 'noptim')
minnumstr	Indicates the minimum number of units that must be allocated in each stratum. Default is 2. This parameter has to be given in a vectorial format, whose length is given by the number of different optimisations (= value of parameter 'noptim')
iter	Indicated the maximum number of iterations (= generations) of the genetic algorithm. Default is 20. This parameter has to be given in a vectorial format, whose length is given by the number of different optimisations (= value of parameter 'noptim')
pops	The dimension of each generations in terms of individuals. Default is 50. This parameter has to be given in a vectorial format, whose length is given by the number of different optimisations (= value of parameter 'noptim')
mut_chance	Mutation chance: for each new individual, the probability to change each single chromosome, i.e. one bit of the solution vector. High values of this parameter allow a deeper exploration of the solution space, but a slower convergence, while low values permit a faster convergence, but the final solution can be distant from the optimal one. Default is 0.05. This parameter has to be given in a vectorial format, whose length is given by the number of different optimisations (= value of parameter 'noptim')
elitism_rate	This parameter indicates the rate of better solutions that must be preserved from one generation to another. Default is 0.2 (20 This parameter has to be given in a vectorial format, whose length is given by the number of different optimisations (= value of parameter 'noptim')

Author(s)

Giulio Barcaroli

Examples

```

#
## Not run:
#-----
# data setting
library(SamplingStrata)
data(swisstrata)
data(swiserrors)
data(swissframe)
# As this function can be applied only to a given domain per time,
# we select the first domain
frame <- swissframe[swissframe$domainvalue == 1,]
strata <- swisstrata[swisstrata$DOM1 == 1,]
errors <- swiserrors[swiserrors$domainvalue == 1,]
#-----
# parameters setting
noptim <- 8 # Number of runs
nsampl <- 100 # Number of samples to be drawn after each optimization
initialStrata <- ceiling(c(1:noptim)*0.1*(nrow(strata))) # Number of initial strata
addStrataFactor <- rep(0.01,noptim) # Rate for increasing initial strata
minnumstr <- rep(2,noptim) # Minimum number of units per stratum
iter <- rep(200,noptim) # Number of iterations for each optimization
pops <- rep(20,noptim) # Number of solutions for each iteration
mut_chance <- rep(0.004,noptim) # Mutation chance
elitism_rate <- rep(0.2,noptim) # Elitism rate
#-----
tuneParameters (
  noptim,
  nsampl,
  frame,
  errors = errors,
  strata = strata,
  cens = NULL,
  strcens = FALSE,
  alldomains = FALSE,
  dom = 1,
  initialStrata,
  addStrataFactor,
  minnumstr,
  iter,
  pops,
  mut_chance,
  elitism_rate
)

## End(Not run)

```

Description

Once optimal stratification has been obtained, and new labels have been attributed to initial atomic strata ("newstrata"), it is important to report the new classification of units in the sampling frame by attributing new strata labels to each unit. By executing this function, a new frame will be obtained with the same structure of the old, but with the addition of a new stratum label. The initial frame must contain a variable named 'domainvalue' that indicates the same values of the domain that has been used with the 'optimizeStrata' function. If no domains have been defined, this variable will contain all 1's, but it must exist

Usage

```
updateFrame(frame, newstrata, writeFiles = FALSE)
```

Arguments

frame	This is the (mandatory) dataframe containing the sampling frame.
newstrata	This is the (mandatory) dataframe containing the information related to the optimisation applied to initial stratification (new labels applied to atomic strata). It is produced by executing the "updateStrata" function.
writeFiles	Flag to write or not the new sampling frame into the working directory. Default is "FALSE"

Value

A dataframe containing the frame

Author(s)

Giulio Barcaroli

Examples

```
#
# The following example is realistic, but is time consuming
#
## Not run:
library(SamplingStrata)
data(swisserrors)
data(swissstrata)
# optimisation of sampling strata
solution <- optimizeStrata (
  errors = swisserrors,
  strata = swissstrata,
  cens = NULL,
  strcens = FALSE,
  initialStrata = 3000,
  addStrataFactor = 0.01,
  minnumstr = 2,
  iter = 60,
  pops = 20,
  mut_chance = 0.05,
  elitism_rate = 0.2,
  highvalue = 100000000,
  suggestions = NULL,
```

```

writeFiles = FALSE)
# updating sampling strata with new strata labels
newstrata <- updateStrata(swissstrata, solution, writeFiles = TRUE)
# updating sampling frame with new strata labels
data(swissframe)
framewnew <- updateFrame(frame=swissframe, newstrata=newstrata, writeFiles = TRUE)

## End(Not run)

```

updateStrata	<i>Assigns new labels to atomic strata on the basis of the optimized aggregated strata</i>
--------------	--

Description

Once optimal stratification has been obtained ('outstrata'), then we need to attribute new strata labels to each atomic stratum. By executing this function, a new dataframe "newstrata" will be obtained with the same structure of the old, ("strata") but with the addition of a new stratum label. By indicating "YES" to "writeFile" parameter, the dataframe "newstrata" will be written to a delimited file ("newstrata.txt"). Also a second delimited file ("strata_aggregation.txt") will be outputted, containing the indication of the relations between atomic and aggregated strata.

Usage

```
updateStrata(strata, solution, writeFiles = FALSE)
```

Arguments

strata	This is the (mandatory) dataframe containing the information related to the atomic strata to which the optimisation has been applied to.
solution	List obtained by the execution of the "optimizeStrata" function. The first element of the list is the vector of the indices corresponding to the optimal solution.
writeFiles	Indicates if at the end of the processing the resulting strata will be outputted in a delimited file. Default is "FALSE".

Value

A dataframe containing the strata

Author(s)

Giulio Barcaroli

Examples

```

#
# This is a toy example, and can be run
#

library(SamplingStrata)
data(errors)
data(strata)

```

```

# optimisation of sampling strata
solution <- optimizeStrata (
  errors = errors,
  strata = strata,
  cens = NULL,
  strcens = FALSE,
  initialStrata = 3000,
  addStrataFactor = 0.01,
  minnumstr = 2,
  iter = 30,
  pops = 20,
  mut_chance = 0.05,
  elitism_rate = 0.2,
  highvalue = 100000000,
  suggestions = NULL,
writeFiles = FALSE)
# updating sampling strata with new strata labels
newstrata <- updateStrata(strata, solution)

#
# The following example is realistic, but is time consuming
#
## Not run:
library(SamplingStrata)
data(swisserrors)
data(swissstrata)
# optimisation of sampling strata
solution <- optimizeStrata (
  errors = swisserrors,
  strata = swissstrata,
  cens = NULL,
  strcens = FALSE,
  initialStrata = 3000,
  addStrataFactor = 0.01,
  minnumstr = 2,
  iter = 60,
  pops = 20,
  mut_chance = 0.05,
  elitism_rate = 0.2,
  highvalue = 100000000,
  suggestions = NULL,
writeFiles = FALSE)
# updating sampling strata with new strata labels
newstrata <- updateStrata(swissstrata, solution)

## End(Not run)

```

var.bin

Allows to transform a continuous variable into a categorical ordinal one by applying a modified version of the k-means clustering function in the 'stats' package.

Description

The optimization of a frame stratification is applicable only in presence of all categorical auxiliary variables in the frame. If one or more continuous auxiliary variables are in the frame, it is necessary

to pre-process in order to convert them into categorical (ordinal) variables. The applied method is the "k-means" clustering method contained in the "stats" package. This function ensures that the final result is in an ordered categorical variable.

Usage

```
var.bin(x,  
bins=3,  
iter.max=100)
```

Arguments

x	Continuous variable to be transformed into a categorical one
bins	Number of values of the resulting categorical variable
iter.max	Maximum number of iterations of the clustering algorithm

Value

Binned variable

Examples

```
library(SamplingStrata)  
data(swissmunicipalities)  
data(swissframe)  
swissframe$X1 <- var.bin(swissmunicipalities$POPTOT,bins = 18)  
table(swissframe$X1)  
tapply(swissmunicipalities$POPTOT,swissframe$X1,mean)
```

Index

*Topic **datasets**

- errors, [5](#)
- strata, [12](#)
- swisserrors, [13](#)
- swissframe, [14](#)
- swissmunicipalities, [15](#)
- swissstrata, [16](#)

*Topic **survey**

- bethel, [2](#)
- buildStrataDF, [3](#)
- checkInput, [4](#)
- evalSolution, [6](#)
- optimizeStrata, [7](#)
- plotSamprate, [10](#)
- selectSample, [11](#)
- tuneParameters, [17](#)
- updateFrame, [19](#)
- updateStrata, [21](#)
- var.bin, [22](#)

[bethel, 2](#)

[buildStrataDF, 3](#)

[checkInput, 4](#)

[errors, 5](#)

[evalSolution, 6](#)

[optimizeStrata, 7](#)

[plotSamprate, 10](#)

[selectSample, 11](#)

[strata, 12](#)

[swisserrors, 13](#)

[swissframe, 14](#)

[swissmunicipalities, 15](#)

[swissstrata, 16](#)

[tuneParameters, 17](#)

[updateFrame, 19](#)

[updateStrata, 21](#)

[var.bin, 22](#)