Package ‘StratifiedSampling’

January 31, 2025

Type Package

Title Different Methods for Stratified Sampling

Version 0.4.2

Description Integrating a stratified structure in the population in a sampling design can considerably reduce the variance of the Horvitz-Thompson estimator. We propose in this package different methods to handle the selection of a balanced sample in stratified population. For more details see Raphaël Jauslin, Esther Eustache and Yves Tillé (2021) <doi:10.1007/s42081-021-00134-y>. The package propose also a method based on optimal transport and balanced sampling, see Raphaël Jauslin and Yves Tillé <doi:10.1016/j.jspi.2022.12.003>.

URL https://github.com/RJauslin/StratifiedSampling

BugReports https://github.com/RJauslin/StratifiedSampling/issues

License GPL (>= 2)

Encoding UTF-8

LinkingTo RcppArmadillo, Rcpp

Imports Rcpp, transport, proxy, MASS, sampling, Rglpk

Depends Matrix, R (>= 3.5.0)

Suggests knitr, rmarkdown, BalancedSampling, stats, testthat, StatMatch, laeken, prettydoc, ggplot2, viridis, geojsonio, sf, rmapshaper

RoxygenNote 7.3.2

VignetteBuilder knitr

NeedsCompilation yes

Author Raphael Jauslin [aut, cre] (<https://orcid.org/0000-0003-1088-3356>),
Esther Eustache [aut],
Bardia Panahbehagh [aut] (<https://orcid.org/0000-0001-9122-7777>),
Yves Tillé [aut] (<https://orcid.org/0000-0003-0904-5523>)

Maintainer Raphael Jauslin <raphael.jauslin@bfs.admin.ch>

Repository CRAN

Date/Publication 2025-01-31 16:20:02 UTC

Contents

`balseq` *Sequential balanced sampling*

Description

Selects at the same time a well-spread and a balanced sample using a sequential implementation.

Usage

```
balseq(pik, Xaux, Xspread = NULL, rord = TRUE)
```

Arguments

- `pik` A vector of inclusion probabilities.
- `Xaux` A matrix of auxiliary variables. The matrix must contains the `pik` vector to have fixed sample size.
- `Xspread` An optional matrix of spatial coordinates.
- `rord` A logical variable that specify if reordering is applied. Default TRUE.

Details

The function selects a sample using a sequential algorithm. At the same time, it respects the balancing equations (`Xaux`) and select a well-spread sample (`Xspread`). Algorithm uses a linear program to satisfy the constraints.

Value

Return the selected indices in 1,2,...,N

See Also

`BalancedSampling:lcube`, `sampling:samplecube`.

Examples

```
N <- 100
n <- 10
p <- 10

pik <- rep(n/N,N)

Xaux <- array(rnorm(N*p,3,1),c(N,p))

Xspread <- cbind(runif(N),runif(N))
Xaux <- cbind(pik,Xaux)

s <- balseq(pik,Xaux)
colSums(Xaux[s,]/as.vector(pik[s]))
colSums(Xaux)


s <- balseq(pik,Xaux,Xspread)
colSums(Xaux[s,]/as.vector(pik[s]))
colSums(Xaux)
```

balstrat — *Balanced Stratification*

Description

Select a stratified balanced sample. The function is similar to balancedstratification of the package sampling.

Usage

```
balstrat(X, strata, pik, rand = TRUE, landing = TRUE)
```

Arguments

X	A matrix of size (N x p) of auxiliary variables on which the sample must be balanced.
strata	A vector of integers that specifies the stratification.
pik	A vector of inclusion probabilities.
rand	if TRUE, the data are randomly arranged. Default TRUE
landing	if TRUE, landing by linear programming otherwise supression of variables. Default TRUE

Details

The function implements the method proposed by Chauvet (2009). Firstly, a flight phase is performed on each strata. Secondly, a flight phase is applied on the whole population by aggregating the strata. Finally, a landing phase is applied by suppression of variables.

Value

A vector with elements equal to 0 or 1. The value 1 indicates that the unit is selected while the value 0 is for rejected units.

References

Chauvet, G. (2009). Stratified balanced sampling. *Survey Methodology*, 35:115-119.

See Also

ffphase, landingRM

Examples

```
N <- 100
n <- 10
p <- 4
X <- matrix(rgamma(N*p,4,25),ncol = p)
strata <- as.matrix(rep(1:n,each = N/n))
pik <- rep(n/N,N)

s <- balstrat(X,strata,pik)

t(X/pik)%*%s
t(X/pik)%*%pik

Xcat <- disj(strata)

t(Xcat)%*%s
t(Xcat)%*%pik
```

---

bsmatch *Statistical matching using optimal transport and balanced sampling*

Description

We propose a method based on the output of the function `otmatch`. The method consists of choosing a unit from sample 2 to assign to a particular unit from sample 1.

Usage

```
bsmatch(object, Z2)
```

Arguments

`object`	A data.frame, output from the function `otmatch`.
`Z2`	A optional matrix, if we want to add some variables for the stratified balanced sampling step.

Details

All details of the method can be seen in the manuscript: Raphaël Jauslin and Yves Tillé (2021) <arXiv:2105.08379>.

Value

A list of two objects, A data.frame that contains the matching and the normalized weights. The first two columns of the data.frame contain the unit identities of the two samples. The third column are the final weights. All remaining columns are the matching variables.

See Also

otmatch, stratifiedcube

Examples

```
#--- SET UP
N=1000
p=5
X=array(rnorm(N*p),c(N,p))
EPS= 1e-9

n1=100
n2=200

s1=sampling::srswor(n1,N)
s2=sampling::srswor(n2,N)


id1=(1:N)[s1==1]
id2=(1:N)[s2==1]

d1=rep(N/n1,n1)
d2=rep(N/n2,n2)

X1=X[s1==1,]
X2=X[s2==1,]

#--- HARMONIZATION

re=harmonize(X1,d1,id1,X2,d2,id2)
w1=re$w1
w2=re$w2

#--- STATISTICAL MATCHING WITH OT

object = otmatch(X1,id1,X2,id2,w1,w2)

#--- BALANCED SAMPLING

out <- bsmatch(object)
```

---

calibRaking *Calibration using raking ratio*

Description

This function is inspired by the function calib of the package sampling. It computes the g-weights of the calibration estimator.

Usage

```
calibRaking(Xs, d, total, q, max_iter = 500L, tol = 1e-09)
```

Arguments

Xs	A matrix of calibration variables.
d	A vector, the initial weights.
total	A vector that represents the initial weights.
q	A vector of positive value that account for heteroscedasticity.
max_iter	An integer, the maximum number of iterations. Default = 500.
tol	A scalar that represents the tolerance value for the algorithm. Default = 1e-9.

Details

More details on the different calibration methods can be read in Tillé Y. (2020).

Value

A vector, the value of the g-weights.

References

Tillé, Y. (2020). *Sampling and estimation from finite populations*. Wiley, New York

---

cps — *Conditional Poisson sampling design*

Description

Maximum entropy sampling with fixed sample size. It select a sample with fixed sample size with unequal inclusion probabilities.

Usage

```
cps(pik, eps = 1e-06)
```

Arguments

pik	A vector of inclusion probabilities.
eps	A scalar that specify the tolerance to transform a small value to the value 0.

Details

Conditional Poisson sampling, the sampling design maximizes the entropy:

$$I(p) = -\sum sp(s)log[p(s)].$$

where s is of fixed sample size. Indeed, Poisson sampling is known for maximizing the entropy but has no fixed sample size. The function selects a sample of fixed sample that maximizes entropy.

This function is a C++ implementation of UPmaxentropy of the package sampling. More details could be find in Tille (2006).

Value

A vector with elements equal to 0 or 1. The value 1 indicates that the unit is selected while the value 0 is for rejected units.

References

Tille, Y. (2006), Sampling Algorithms, springer

Examples

```
pik <- inclprob(seq(100,1,length.out = 100),10)
s <-  cps(pik)



# simulation with piktfrompik MUCH MORE FASTER
s <- rep(0,length(pik))
SIM <- 100
pikt <- piktfrompik(pik)
w <- pikt/(1-pikt)
q <- qfromw(w,sum(pik))
for(i in 1 :SIM){
  s <- s + sfromq(q)
}
p <- s/SIM # estimated inclusion probabilities
t <- (p-pik)/sqrt(pik*(1-pik)/SIM)
1 - sum(t > 1.6449)/length(pik) # should be approximately equal to 0.95
```

---

c_bound *C bound*

Description

This function is returning the number of unit that we need such that some conditions are fulfilled. See Details

Usage

```
c_bound(pik)
```

Arguments

pik	vector of the inclusion probabilities.

Details

The function is computing the number of unit K that we need to add such that the following conditions are fulfilled :

- $\sum_{k=1}^K \pi_k \geq 1$
- $\sum_{k=1}^K 1 - \pi_k \geq 1$
- Let c be the constant such that $\sum_{k=2}^K min(c\pi_k, 1) = n$, we must have that $\pi_1 \geq 1 - 1/c$

Value

An integer value, the number of units that we need to respect the constraints.

See Also

osod

---

c_bound2 *C bound*

Description

This function is returning the number of unit that we need such that some conditions are fulfilled. See Details

Usage

```
c_bound2(pik)
```

Arguments

pik	vector of the inclusion probabilities.

Details

The function is computing the number of unit K that we need to add such that the following conditions are fulfilled :

- $\sum_{k=1}^K \pi_k \geq 1$
- $\sum_{k=1}^K 1 - \pi_k \geq 1$
- Let c be the constant such that $\sum_{k=2}^K min(c\pi_k, 1) = n$, we must have that $\pi_1 \geq 1 - 1/c$

Value

An integer value, the number of units that we need to respect the constraints.

See Also

osod

---

disj *Disjunctive*

Description

This function transforms a categorical vector into a matrix of indicators.

Usage

```
disj(strata_input)
```

Arguments

strata_input A vector of integers that represents the categories.

Value

A matrix of indicators.

Examples

```
strata <- rep(c(1,2,3),each = 4)
disj(strata)
```

---

disjMatrix *Disjunctive for matrix*

Description

This function transforms a categorical matrix into a matrix of indicators variables.

Usage

```
disjMatrix(strata_input)
```

Arguments

strata_input A matrix of integers that contains categorical vector in each column.

Value

A matrix of indicators.

Examples

```
Xcat <-  matrix(c(sample(x = 1:6, size = 100, replace = TRUE),
            sample(x = 1:6, size = 100, replace = TRUE),
            sample(x = 1:6, size = 100, replace = TRUE)),ncol = 3)
disjMatrix(Xcat)
```

---

distUnitk — *Squared Euclidean distances of the unit k.*

Description

Calculate the squared Euclidean distance from unit k to the other units.

Usage

```
distUnitk(X, k, tore, toreBound)
```

Arguments

`X`	matrix representing the spatial coordinates.
`k`	the unit index to be used.
`tore`	an optional logical value, if we are considering the distance on a tore. See Details.
`toreBound`	an optional numeric value that specify the length of the tore.

Details

Let $\mathbf{x}_k, \mathbf{x}_l$ be the spatial coordinates of the unit $k, l \in U$. The classical euclidean distance is given by

$$d^2(k,l) = (\mathbf{x}_k - \mathbf{x}_l)^\top (\mathbf{x}_k - \mathbf{x}_l).$$

When the points are distributed on a $N_1 \times N_2$ regular grid of R^2. It is possible to consider the units like they were placed on a tore. It can be illustrated by Pac-Man passing through the wall to get away from ghosts. Specifically, we could consider two units on the same column (resp. row) that are on the opposite have a small distance,

$$d_T^2(k,l) = min((x_{k_1} - x_{l_1})^2, (x_{k_1} + N_1 - x_{l_1})^2, (x_{k_1} - N_1 - x_{l_1})^2) + \\ min((x_{k_2} - x_{l_2})^2, (x_{k_2} + N_2 - x_{l_2})^2, (x_{k_2} - N_2 - x_{l_2})^2).$$

The option `toreBound` specify the length of the tore in the case of $N_1 = N_2 = N$. It is omitted if the `tore` option is equal to `FALSE`.

Value

a vector of length N that contains the distances from the unit k to all other units.

See Also

dist.

Examples

```
N <- 5
x <- seq(1,N,1)
X <- as.matrix(expand.grid(x,x))
distUnitk(X,k = 2,tore = TRUE,toreBound = 5)
distUnitk(X,k = 2,tore = FALSE,toreBound = -1)
```

---

fbs *Fast Balanced Sampling*

Description

This function implements the method proposed by Hasler and Tillé (2014). It should be used for selecting a sample from highly stratified population.

Usage

```
fbs(X, strata, pik, rand = TRUE, landing = TRUE)
```

Arguments

`X`	A matrix of size (N x p) of auxiliary variables on which the sample must be balanced.
`strata`	A vector of integers that specifies the stratification.
`pik`	A vector of inclusion probabilities.
`rand`	if TRUE, the data are randomly arranged. Default TRUE
`landing`	if TRUE, landing by linear programming otherwise supression of variables. Default TRUE

Details

Firstly a flight phase is performed on each strata. Secondly, several flight phases are applied by adding one by one the stratum. By doing this, some strata are managed on-the-fly. Finally, a landing phase is applied by suppression of the variables. If the number of element selected in each stratum is not equal to an integer, the function can be very time-consuming.

Value

A vector with elements equal to 0 or 1. The value 1 indicates that the unit is selected while the value 0 is for rejected units.

References

Hasler, C. and Tillé Y. (2014). Fast balanced sampling for highly stratified population. *Computational Statistics and Data Analysis*, 74, 81-94

Examples

```
N <- 100
n <- 10
x1 <- rgamma(N,4,25)
x2 <- rgamma(N,4,25)

strata <- rep(1:n,each = N/n)

pik <- rep(n/N,N)
X <- as.matrix(cbind(matrix(c(x1,x2),ncol = 2)))

s <- fbs(X,strata,pik)

t(X/pik)%*%s
t(X/pik)%*%pik

Xcat <- disj(strata)

t(Xcat)%*%s
t(Xcat)%*%pik
```

`ffphase` *Fast flight phase of the cube method*

Description

This function computes the flight phase of the cube method proposed by Chauvet and Tillé (2006).

Usage

```
ffphase(Xbal, prob, order = TRUE)
```

Arguments

`Xbal`	A matrix of size (N x p) of auxiliary variables on which the sample must be balanced.
`prob`	A vector of inclusion probabilities.
`order`	if the units are reordered, Default TRUE.

Details

This function implements the method proposed by (Chauvet and Tillé 2006). It recursively transforms the vector of inclusion probabilities pik into a sample that respects the balancing equations. The algorithm stops when the null space of the sub-matrix B is empty. For more information see (Chauvet and Tillé 2006).

Value

Updated vector of pik that contains 0 and 1 for unit that are rejected or selected.

See Also

fastflightphase, cube.

Examples

```
N <- 100
n <- 10
p <- 4
pik <- rep(n/N,N)
X <- cbind(pik,matrix(rgamma(N*p,4,25),ncol= p))

pikstar <- ffphase(X,pik)
t(X/pik)%*%pikstar
t(X/pik)%*%pik
pikstar
```

---

findB *Find best sub-matrix B in stratifiedcube*

Description

This function is computing a sub-matrix used in stratifiedcube.

Usage

```
findB(X, strata)
```

Arguments

X	A matrix of size (N x p) of auxiliary variables on which the sample must be balanced.
strata	A vector of integers that specifies the stratification.

Details

The function finds the smallest matrix B such that it contains only one more row than the number of columns. It consecutively adds the right number of rows depending on the number of categories that is added.

Value

A list of two components. The sub-matrix of X and the corresponding disjunctive matrix. If we use the function cbind to combine the two matrices, the resulting matrix has only one more row than the number of columns.

Examples

```
N <- 1000
strata <-  sample(x = 1:6, size = N, replace = TRUE)

p <- 3
X <- matrix(rnorm(N*p),ncol = 3)
findB(X,strata)
```

---

gencalibRaking *Generalized calibration using raking ratio*

Description

This function is inspired by the function calib of the package sampling. It computes the g-weights of the calibration estimator.

Usage

```
gencalibRaking(Xs, Zs, d, total, q, max_iter = 500L, tol = 1e-09)
```

Arguments

Xs	A matrix of calibration variables.
Zs	A matrix of instrumental variables with same dimension as Xs.
d	A vector, the initial weights.
total	A vector that represents the initial weights.
q	A vector of positive value that account for heteroscedasticity.
max_iter	An integer, the maximum number of iterations. Default = 500.
tol	A scalar that represents the tolerance value for the algorithm. Default = 1e-9.

Details

More details on the different calibration methods can be read in Tillé Y. (2020).

Value

A vector, the value of the g-weights.

References

Tillé, Y. (2020). *Sampling and estimation from finite populations*. Wiley, New York

`harmonize` *Harmonization by calibration*

Description

This function harmonize the two weight schemes such that the totals are equal.

Usage

```
harmonize(X1, d1, id1, X2, d2, id2, totals)
```

Arguments

`X1`	A matrix, the matching variables of sample 1.
`d1`	A numeric vector that contains the initial weights of the sample 1.
`id1`	A character or numeric vector that contains the labels of the units in sample 1.
`X2`	A matrix, the matching variables of sample 2.
`d2`	A numeric vector that contains the initial weights of the sample 1.
`id2`	A character or numeric vector that contains the labels of the units in sample 2.
`totals`	An optional numeric vector that contains the totals of the matching variables.

Details

All details of the method can be seen in the manuscript: Raphaël Jauslin and Yves Tillé (2021) <arXiv:>.

Value

A list of two vectors, the new weights of sample 1 (respectively new weights of sample 2).

Examples

```
#--- SET UP

N = 1000
p = 5
X = array(rnorm(N*p),c(N,p))

n1=100
n2=200

s1 = sampling::srswor(n1,N)
s2 = sampling::srswor(n2,N)

id1=(1:N)[s1==1]
id2=(1:N)[s2==1]
```

```
d1=rep(N/n1,n1)
d2=rep(N/n2,n2)

X1 = X[s1==1,]
X2 = X[s2==1,]

re <- harmonize(X1,d1,id1,X2,d2,id2)

colSums(re$w1*X1)
colSums(re$w2*X2)

#---  if the true totals is known

totals <- c(N,colSums(X))
re <- harmonize(X1,d1,id1,X2,d2,id2,totals)

colSums(re$w1*X1)
colSums(re$w2*X2)
colSums(X)
```

---

inclprob *Inclusion Probabilities*

Description

Computes first-order inclusion probabilities from a vector of positive numbers.

Usage

```
inclprob(x, n)
```

Arguments

x	vector of positive numbers.
n	sample size (could be a positive real value).

Details

The function is implemented in C++ so that it can be used in the code of other C++ functions. The implementation is based on the function `inclusionprobabilities` of the package sampling.

Value

A vector of inclusion probabilities proportional to `x` and such that the sum is equal to the value `n`.

See Also

`inclusionprobabilities`

Examples

```
x <- runif(100)
pik <- inclprob(x,70)
sum(pik)
```

---

landingRM *Landing by suppression of variables*

Description

This function performs the landing phase of the cube method using suppression of variables proposed by Chauvet and Tillé (2006).

Usage

```
landingRM(A, pikstar, EPS = 1e-07)
```

Arguments

A	matrix of auxiliary variables on which the sample must be balanced. (The matrix should be divided by the original inclusion probabilities.)
pikstar	vector of updated inclusion probabilities by the flight phase. See ffphase
EPS	epsilon value

Value

A vector with elements equal to 0 or 1. The value 1 indicates that the unit is selected while the value 0 is for rejected units.

References

Chauvet, G. and Tillé, Y. (2006). A fast algorithm of balanced sampling. *Computational Statistics*, 21/1:53-62

See Also

fbs, balstrat.

Examples

```
N <- 1000
n <- 10
p <- 4
pik <- rep(n/N,N)
X <- cbind(pik,matrix(rgamma(N*p,4,25),ncol= p))
pikstar <- ffphase(X,pik)
s <- landingRM(X/pik*pikstar,pikstar)
```

```
sum(s)
t(X/pik)%*%pik
t(X/pik)%*%pikstar
t(X/pik)%*%s
```

---

maxentpi2 *Joint inclusion probabilities of maximum entropy.*

Description

This function computes the matrix of the joint inclusion of the maximum entropy sampling with fixed sample size. It can handle unequal inclusion probabilities.

Usage

```
maxentpi2(pikr)
```

Arguments

pikr	A vector of inclusion probabilities.

Details

The sampling design maximizes the entropy design:

$$I(p) = -\sum sp(s)log[p(s)].$$

This function is a C++ implementation of UPMEpik2frompikw. More details could be find in Tille (2006).

Value

A matrix, the joint inclusion probabilities.

References

Tille, Y. (2006), Sampling Algorithms, springer

---

ncat *Number of categories*

Description

This function returns the number of factor in each column of a categorical matrix.

Usage

```
ncat(Xcat_input)
```

Arguments

Xcat_input	A matrix of integers that contains categorical vector in each column.

Value

A row vector that contains the number of categories in each column.

Examples

```
Xcat <-  matrix(c(sample(x = 1:6, size = 100, replace = TRUE),
            sample(x = 1:6, size = 100, replace = TRUE),
            sample(x = 1:6, size = 100, replace = TRUE)),ncol = 3)
ncat(Xcat)
```

---

osod *One-step One Decision sampling method*

Description

This function implements the One-step One Decision method. It can be used using equal or unequal inclusion probabilities. The method is particularly useful for selecting a sample from a stream.

Usage

```
osod(pikr, full = FALSE)
```

Arguments

pikr	A vector of inclusion probabilities.
full	An optional boolean value, to specify whether the full population (the entire vector) is used to update inclusion probabilities. Default: FALSE

Details

The method sequentially transforms the vector of inclusion probabilities into a sample whose values are equal to 0 or 1. The method respects the inclusion probabilities and can handle equal or unequal inclusion probabilities.

The method does not take into account the whole vector of inclusion probabilities by having a sequential implementation. This means that the method is fast and can be implemented in a flow.

Value

A vector with elements equal to 0 or 1. The value 1 indicates that the unit is selected while the value 0 is for rejected units.

See Also

c_bound

Examples

```
N <- 1000
n <- 100
pik <- inclprob(runif(N),n)
s <- osod(pik)
```

---

otmatch — *Statistical Matching using Optimal transport*

Description

This function computes the statistical matching between two complex survey samples with weighting schemes. The function uses the function `transport` of the package `transport`.

Usage

```
otmatch(
  X1,
  id1,
  X2,
  id2,
  w1,
  w2,
  dist_method = "Euclidean",
  transport_method = "shortsimplex",
  EPS = 1e-09
)
```

Arguments

- `X1` A matrix, the matching variables of sample 1.
- `id1` A character or numeric vector that contains the labels of the units in sample 1.
- `X2` A matrix, the matching variables of sample 2.
- `id2` A character or numeric vector that contains the labels of the units in sample 1.
- `w1` A numeric vector that contains the weights of the sample 1, harmonized by the function `harmonize`.
- `w2` A numeric vector that contains the weights of the sample 2, harmonized by the function `harmonize`.
- `dist_method` A string that specified the distance used by the function `dist` of the package `proxy`. Default `"Euclidean"`.
- `transport_method` A string that specified the distance used by the function `transport` of the package `transport`. Default `"shortsimplex"`.
- `EPS` an numeric scalar to determine if the value is rounded to 0.

Details

All details of the method can be seen in : Raphaël Jauslin and Yves Tillé (2021) <arXiv:2105.08379>.

Value

A data.frame that contains the matching. The first two columns contain the unit identities of the two samples. The third column is the final weights. All remaining columns are the matching variables.

Examples

```
#--- SET UP
N=1000
p=5
X=array(rnorm(N*p),c(N,p))
EPS= 1e-9

n1=100
n2=200

s1 = sampling::srswor(n1,N)
s2 = sampling::srswor(n2,N)


id1=(1:N)[s1==1]
id2=(1:N)[s2==1]

d1=rep(N/n1,n1)
d2=rep(N/n2,n2)

X1=X[s1==1,]
X2=X[s2==1,]
```

```
#--- HARMONIZATION

re=harmonize(X1,d1,id1,X2,d2,id2)
w1=re$w1
w2=re$w2

#--- STATISTICAL MATCHING WITH OT

object = otmatch(X1,id1,X2,id2,w1,w2)


round(colSums(object$weight*object[,4:ncol(object)]),3)
round(colSums(w1*X1),3)
round(colSums(w2*X2),3)
```

`pikfromq` *pik from q*

Description

This function finds the `pik` from an initial `q`.

Usage

```
pikfromq(q)
```

Arguments

`q`	A matrix that is computed from the function `qfromw`.

Details

More details could be find in Tille (2006).

Value

A vector of inclusion probability computed from the matrix q.

References

Tille, Y. (2006), Sampling Algorithms, springer

`piktfrompik` *pikt from pik*

Description

This function finds the `pikt` from an initial `pik`.

Usage

```
piktfrompik(pik, max_iter = 500L, tol = 1e-08)
```

Arguments

`pik`	A vector of inclusion probabilities. The vector must contains only value that are not integer.
`max_iter`	An integer that specify the maximum iteration in the Newton-Raphson algorithm. Default `500`.
`tol`	A scalar that specify the tolerance convergence for the Newton-Raphson algorithm. Default `1e-8`.

Details

The management of probabilities equal to 0 or 1 is done in the cps function.

`pikt` is the vector of inclusion probabilities of a Poisson sampling with the right parameter. The vector is found by Newtwon-Raphson algorithm.

More details could be find in Tille (2006).

Value

An updated vector of inclusion probability.

References

Tille, Y. (2006), Sampling Algorithms, springer

qfromw — *q from w*

Description

This function finds the matrix q form a particular w.

Usage

```
qfromw(w, n)
```

Arguments

w	A vector of weights.
n	An integer that is equal to the sum of the inclusion probabilities.

Details

w is generally computed by the formula pik/(1-pik), where n is equal to the sum of the vector pik. More details could be find in Tille (2006).

Value

A matrix of size N x n, where N is equal to the length of the vector w.

References

Tille, Y. (2006), Sampling Algorithms, springer

sfromq — *s from q*

Description

This function finds sample s form the matrix q.

Usage

```
sfromq(q)
```

Arguments

q	A matrix that is computed from the function qfromw.

Details

More details could be find in Tille (2006).

Value

A vector with elements equal to 0 or 1. The value 1 indicates that the unit is selected while the value 0 is for rejected units.

References

Tille, Y. (2006), Sampling Algorithms, springer

`stratifiedcube` *Stratified Sampling*

Description

This function implements a method for selecting a stratified sample. It really improves the performance of the function `fbs` and `balstrat`.

Usage

```
stratifiedcube(
  X,
  strata,
  pik,
  EPS = 1e-07,
  rand = TRUE,
  landing = TRUE,
  lp = TRUE
)
```

Arguments

`X`	A matrix of size (N x p) of auxiliary variables on which the sample must be balanced.
`strata`	A vector of integers that specifies the stratification..
`pik`	A vector of inclusion probabilities.
`EPS`	epsilon value
`rand`	if TRUE, the data are randomly arranged. Default TRUE
`landing`	if FALSE, no landing phase is done.
`lp`	if TRUE, landing by linear programming otherwise supression of variables. Default TRUE

Details

The function is selecting a balanced sample very quickly even if the sum of inclusion probabilities within strata are non-integer. The function should be used in preference. Firstly, a flight phase is performed on each strata. Secondly, the function `findB` is used to find a particular matrix to apply a flight phase by using the cube method proposed by Chauvet, G. and Tillé, Y. (2006). Finally, a landing phase is applied by suppression of variables.

Value

A vector with elements equal to 0 or 1. The value 1 indicates that the unit is selected while the value 0 is for rejected units.

References

Chauvet, G. and Tillé, Y. (2006). A fast algorithm of balanced sampling. *Computational Statistics*, 21/1:53-62

See Also

fbs, balstrat, landingRM, ffphase

Examples

```
# EXAMPLE WITH EQUAL INCLUSION PROBABILITES AND SUM IN EACH STRATA INTEGER
N <- 100
n <- 10
p <- 4
X <- matrix(rgamma(N*p,4,25),ncol = p)
strata <- rep(1:n,each = N/n)
pik <- rep(n/N,N)

s <- stratifiedcube(X,strata,pik)

t(X/pik)%*%s
t(X/pik)%*%pik

Xcat <- disj(strata)

t(Xcat)%*%s
t(Xcat)%*%pik


# EXAMPLE WITH UNEQUAL INCLUSION PROBABILITES AND SUM IN EACH STRATA INTEGER
N <- 100
n <- 10
X <- cbind(rgamma(N,4,25),rbinom(N,20,0.1),rlnorm(N,9,0.1),runif(N))
colSums(X)
strata <- rbinom(N,10,0.7)
strata <- sampling::cleanstrata(strata)
pik <- as.vector(sampling::inclusionprobastrata(strata,ceiling(table(strata)*0.10)))
EPS = 1e-7

s <- stratifiedcube(X,strata,pik)
test <- stratifiedcube(X,strata,pik,landing = FALSE)

t(X/pik)%*%s
t(X/pik)%*%test
t(X/pik)%*%pik

Xcat <- disj(strata)
```

```
t(Xcat)%*%s
t(Xcat)%*%test
t(Xcat)%*%pik


# EXAMPLE WITH UNEQUAL INCLUSION PROBABILITES AND SUM IN EACH STRATA NOT INTEGER
set.seed(3)
N <- 100
n <- 10
X <- cbind(rgamma(N,4,25),rbinom(N,20,0.1),rlnorm(N,9,0.1),runif(N))
strata <- rbinom(N,10,0.7)
strata <- sampling::cleanstrata(strata)
pik <- runif(N)
EPS = 1e-7
tapply(pik,strata,sum)
table(strata)


s <- stratifiedcube(X,strata,pik,landing = TRUE)
test <- stratifiedcube(X,strata,pik,landing = FALSE)


t(X/pik)%*%s
t(X/pik)%*%test
t(X/pik)%*%pik

Xcat <- disj(strata)

t(Xcat)%*%s
t(Xcat)%*%pik
t(Xcat)%*%test
```

`sys_deville` *Deville's systematic*

Description

This function implements a method to select a sample using the Deville's systmatic algorithm.

Usage

```
sys_deville(pik)
```

Arguments

`pik`	A vector of inclusion probabilities.

Value

Return the selected indices in 1,2,...,N

References

Deville, J.-C. (1998), Une nouvelle méthode de tirage à probabilité inégales. Technical Report 9804, Ensai, France.

Chauvet, G. (2012), On a characterization of ordered pivotal sampling, Bernoulli, 18(4):1320-1340

Examples

```
set.seed(1)
pik <- c(0.2,0.5,0.3,0.4,0.9,0.8,0.5,0.4)
sys_deville(pik)
```

---

sys_devillepi2 *Second order inclusion probabilities of Deville's systematic*

Description

This function returns the second order inclusion probabilities of Deville's systematic.

Usage

```
sys_devillepi2(pik)
```

Arguments

pik	A vector of inclusion probabilities

Value

A matrix of second order inclusion probabilities.

References

Deville, J.-C. (1998), Une nouvelle méthode de tirage à probabilité inégales. Technical Report 9804, Ensai, France.

Chauvet, G. (2012), On a characterization of ordered pivotal sampling, Bernoulli, 18(4):1320-1340

Examples

```
set.seed(1)
N <- 30
n <- 4
pik <- as.vector(inclprob(runif(N),n))
PI <- sys_devillepi2(pik)
#image(as(as.matrix(PI),"sparseMatrix"))

pik <- c(0.2,0.5,0.3,0.4,0.9,0.8,0.5,0.4)
PI <- sys_devillepi2(pik)
#image(as(as.matrix(PI),"sparseMatrix"))
```

---

vApp *Approximated variance for balanced sample*

Description

Variance approximation calculated as the conditional variance with respect to the balancing equations of a particular Poisson design. See Tillé (2020)

Usage

```
vApp(Xaux, pik, y)
```

Arguments

Xaux	A matrix of size (N x p) of auxiliary variables on which the sample must be balanced.
pik	A vector of inclusion probabilities. The vector has the size N of the population U.
y	A variable of interest.

Value

Approximated variance of the Horvitz-Thompson estimator.

References

Tillé, Y. (2020), Sampling and Estimation from finite populations, Wiley,

See Also

vDBS vApp

Examples

```
N <- 100
n <- 40
x1 <- rgamma(N,4,25)
x2 <- rgamma(N,4,25)

pik <- rep(n/N,N)
Xaux <- cbind(pik,as.matrix(matrix(c(x1,x2),ncol = 2)))
Xspread <- cbind(runif(N),runif(N))


s <- balseq(pik,Xaux,Xspread)

y <- Xaux%*%c(1,1,3) + rnorm(N,120) # variable of interest

vEst(Xaux[s,],pik[s],y[s])
vDBS(Xaux[s,],Xspread[s,],pik[s],y[s])
vApp(Xaux,pik,y)
```

`varApp` *Approximated variance for balanced sampling*

Description

Approximated variance for balanced sampling

Usage

```
varApp(X, strata, pik, y)
```

Arguments

`X`	A matrix of size (N x p) of auxiliary variables on which the sample must be balanced.
`strata`	A vector of integers that represents the categories.
`pik`	A vector of inclusion probabilities.
`y`	A variable of interest.

Details

This function gives an approximation of the variance of the Horvitz-Thompson total estimator presented by Hasler and Tillé (2014).

Value

a scalar, the value of the approximated variance.

References

Hasler, C. and Tillé, Y. (2014). Fast balanced sampling for highly stratified population. *Computational Statistics and Data Analysis*, 74:81-94.

See Also

varEst

Examples

```
N <- 1000
n <- 400
x1 <- rgamma(N,4,25)
x2 <- rgamma(N,4,25)

strata <- as.matrix(rep(1:40,each = 25)) # 25 strata
Xcat <- disjMatrix(strata)
pik <- rep(n/N,N)
X <- as.matrix(matrix(c(x1,x2),ncol = 2))

s <- stratifiedcube(X,strata,pik)

y <- 20*strata + rnorm(1000,120) # variable of interest
# y_ht <- sum(y[which(s==1)]/pik[which(s == 1)]) # Horvitz-Thompson estimator
# (sum(y_ht) - sum(y))^2 # true variance
varEst(X,strata,pik,s,y)
varApp(X,strata,pik,y)
```

---

varEst *Estimator of the approximated variance for balanced sampling*

Description

Estimator of the approximated variance for balanced sampling

Usage

```
varEst(X, strata, pik, s, y)
```

Arguments

X	A matrix of size (N x p) of auxiliary variables on which the sample must be balanced.
strata	A vector of integers that represents the categories.
pik	A vector of inclusion probabilities.
s	A sample (vector of 0 and 1, if rejected or selected).
y	A variable of interest.

Details

This function gives an estimator of the approximated variance of the Horvitz-Thompson total estimator presented by Hasler C. and Tillé Y. (2014).

Value

a scalar, the value of the estimated variance.

Author(s)

Raphaël Jauslin <raphael.jauslin@unine.ch>

References

Hasler, C. and Tillé, Y. (2014). Fast balanced sampling for highly stratified population. *Computational Statistics and Data Analysis*, 74:81-94.

See Also

varApp

Examples

```
N <- 1000
n <- 400
x1 <- rgamma(N,4,25)
x2 <- rgamma(N,4,25)

strata <- as.matrix(rep(1:40,each = 25)) # 25 strata
Xcat <- disjMatrix(strata)
pik <- rep(n/N,N)
X <- as.matrix(matrix(c(x1,x2),ncol = 2))

s <- stratifiedcube(X,strata,pik)

y <- 20*strata + rnorm(1000,120) # variable of interest
# y_ht <- sum(y[which(s==1)]/pik[which(s == 1)]) # Horvitz-Thompson estimator
# (sum(y_ht) - sum(y))^2 # true variance
varEst(X,strata,pik,s,y)
varApp(X,strata,pik,y)
```

---

vDBS *Variance Estimation for Doubly Balanced Sample.*

Description

Variance estimator for sample that are at the same time well spread and balanced on auxiliary variables. See Grafstr\"om and Till\'e (2013)

Usage

```
vDBS(Xauxs, Xspreads, piks, ys)
```

Arguments

Xauxs	A matrix of size (n x p) of auxiliary variables on which the sample must be balanced.
Xspreads	Matrix of spatial coordinates.
piks	A vector of inclusion probabilities. The vector has the size n of the sample s.
ys	A variable of interest. The vector has the size n of the sample s.

Value

Estimated variance of the horvitz-thompson estimator.

References

Grafstr\"om, A. and Till\'e, Y. (2013), Doubly balanced spatial sampling with spreading and restitution of auxiliary totals, Environmetrics, 14(2):120-131

See Also

vDBS vApp

Examples

```
N <- 100
n <- 40
x1 <- rgamma(N,4,25)
x2 <- rgamma(N,4,25)

pik <- rep(n/N,N)
Xaux <- cbind(pik,as.matrix(matrix(c(x1,x2),ncol = 2)))
Xspread <- cbind(runif(N),runif(N))


s <- balseq(pik,Xaux,Xspread)

y <- Xaux%*%c(1,1,3) + rnorm(N,120) # variable of interest

vEst(Xaux[s,],pik[s],y[s])
vDBS(Xaux[s,],Xspread[s,],pik[s],y[s])
vApp(Xaux,pik,y)
```

vEst *Variance Estimation for balanced sample*

Description

Estimated variance approximation calculated as the conditional variance with respect to the balancing equations of a particular Poisson design. See Tillé (2020)

Usage

```
vEst(Xauxs, piks, ys)
```

Arguments

Xauxs	A matrix of size (n x p) of auxiliary variables on which the sample must be balanced.
piks	A vector of inclusion probabilities. The vector has the size of the sample s.
ys	A variable of interest.The vector has the size n of the sample s.

Value

Estimated variance of the horvitz-thompson estimator.

References

Tillé, Y. (2020), Sampling and Estimation from finite populations, Wiley,

See Also

vDBS vApp

Examples

```
N <- 100
n <- 40
x1 <- rgamma(N,4,25)
x2 <- rgamma(N,4,25)

pik <- rep(n/N,N)
Xaux <- cbind(pik,as.matrix(matrix(c(x1,x2),ncol = 2)))
Xspread <- cbind(runif(N),runif(N))


s <- balseq(pik,Xaux,Xspread)

y <- Xaux%*%c(1,1,3) + rnorm(N,120) # variable of interest

vEst(Xaux[s,],pik[s],y[s])
```

```
vDBS(Xaux[s,],Xspread[s,],pik[s],y[s])
vApp(Xaux,pik,y)
```

Index