

# Package ‘sampling’

February 15, 2012

**Version** 2.5

**Date** 2012-02-09

**Title** Survey Sampling

**Author** Yves Tillé <yves.tille@unine.ch>, Alina Matei <alina.matei@unine.ch>

**Maintainer** Alina Matei <alina.matei@unine.ch>

**Description** Functions for drawing and calibrating samples.

**Imports** lpSolve

**Depends** MASS, lpSolve

**License** GPL (>= 2)

**Encoding** latin1

**Repository** CRAN

**Date/Publication** 2012-02-13 12:05:19

## R topics documented:

balancedcluster . . . . .	3
balancedstratification . . . . .	4
balancedtwostage . . . . .	5
belgianmunicipalities . . . . .	7
calib . . . . .	8
calibev . . . . .	10
checkcalibration . . . . .	12
cleanstrata . . . . .	13
cluster . . . . .	14
disjunctive . . . . .	15
fastflightcube . . . . .	16
gencalib . . . . .	17
getdata . . . . .	19

Hajkeestimator	20
Hajekstrata	21
HTestimator	22
HTstrata	23
inclusionprobabilities	24
inclusionprobastrata	25
landingcube	26
mstage	27
MU284	29
postest	30
poststrata	33
ratioest	34
ratioest_strata	35
regist	37
regist_strata	38
rhg	40
rhg_strata	41
rmodel	42
samplecube	44
srswor	46
srswor1	47
srswr	48
strata	49
swissmunicipalities	51
UPbrewer	52
UPmaxentropy	53
UPmidzuno	55
UPmidzunopi2	56
UPminimalsupport	57
UPmultinomial	58
UPopips	59
UPpivotal	60
UPpoisson	61
UPrandompivotal	62
UPrandomsystematic	63
UPsampford	64
UPsampfordpi2	65
UPsystematic	66
UPsystematicpi2	67
UPtille	68
UPtillepi2	69
varest	70
vartaylor_ratio	71
writesample	72

---

balancedcluster	<i>Balanced cluster</i>
-----------------	-------------------------

---

## Description

Selects a balanced cluster sample.

## Usage

```
balancedcluster(X,m,cluster,selection=1,comment=TRUE,method=1)
```

## Arguments

X	matrix of auxiliary variables on which the sample must be balanced.
m	number of clusters to be selected.
cluster	vector of integers that defines the clusters.
selection	1, selection of the clusters with probabilities proportional to size, 2, selection of the clusters with equal probabilities.
comment	a comment is written during the execution if comment is TRUE.
method	the used method in the function <code>samplecube</code> .

## Value

Returns a matrix containing the vector of inclusion probabilities and the selected sample.

## See Also

[samplecube](#), [fastflightcube](#), [landingcube](#)

## Examples

```
#####
## Example 1
#####
# definition of the clusters; there are 15 units in 3 clusters
cluster=c(1,1,1,1,1,2,2,2,2,2,3,3,3,3,3)
# matrix of balancing variables
X=cbind(c(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15))
# selection of 2 clusters
s=balancedcluster(X,2,cluster,2,TRUE)
# the sample of clusters with the inclusion probabilities of the clusters
s
# the selected clusters
unique(cluster[s[,1]==1])
# the selected units
(1:length(cluster))[s[,1]==1]
# with the probabilities
s[s[,1]==1,2]
```

```
#####
## Example 2
#####
data(MU284)
X=cbind(MU284$P75,MU284$CS82,MU284$SS82,MU284$S82,MU284$ME84)
s=balancedcluster(X,10,MU284$CL,1,TRUE)
cluster=MU284$CL
# the selected clusters
unique(cluster[s[,1]==1])
# the selected units
(1:length(cluster))[s[,1]==1]
# with the probabilities
s[s[,1]==1,2]
```

---

balancedstratification

*Balanced stratification*

---

### Description

Selects a stratified balanced sample (a vector of 0 and 1). Firstly, the flight phase is applied in each stratum. Secondly, the strata are aggregated and the flight phase is applied on the whole population. Finally, the landing phase is applied on the whole population.

### Usage

```
balancedstratification(X,strata,pik,comment=TRUE,method=1)
```

### Arguments

X	matrix of auxiliary variables on which the sample must be balanced.
strata	vector of integers that specifies the stratification.
pik	vector of inclusion probabilities.
comment	a comment is written during the execution if comment is TRUE.
method	the used method in the function samplecube.

### References

Tillé, Y. (2006), *Sampling Algorithms*, Springer.

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

Chauvet, G. and Tillé, Y. (2005). New SAS macros for balanced sampling. In INSEE, editor, *Journées de Méthodologie Statistique*, Paris.

Deville, J.-C. and Tillé, Y. (2004). Efficient balanced sampling: the cube method. *Biometrika*, 91:893–912.

Deville, J.-C. and Tillé, Y. (2005). Variance approximation under balanced sampling. *Journal of Statistical Planning and Inference*, 128/2:411–425.

**See Also**

[samplecube](#), [fastflightcube](#), [landingcube](#)

**Examples**

```
#####
## Example 1
#####
# variable of stratification (3 strata)
strata=c(1,1,1,1,1,2,2,2,2,3,3,3,3)
# matrix of balancing variables
X=cbind(c(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15))
# Vector of inclusion probabilities.
# the sample has its size equal to 9.
pik=rep(3/5,times=15)
# selection of a stratified sample
s=balancedstratification(X,strata,pik,comment=TRUE)
# the sample is
(1:length(pik))[s==1]
#####
## Example 2
#####
data(MU284)
X=cbind(MU284$P75,MU284$CS82,MU284$SS82,MU284$S82,MU284$ME84)
strata=MU284$REG
pik=inclusionprobabilities(MU284$P75,80)
s=balancedstratification(X,strata,pik,TRUE)
#####
## Example 3
#####
data(swissmunicipalities)
swiss=swissmunicipalities
X=cbind(swiss$HApoly,
        swiss$Surfacesbois,
        swiss$P00BMTOT,
        swiss$P00BWTOT,
        swiss$POPTOT,
        swiss$Pop020,
        swiss$Pop2040,
        swiss$Pop4065,
        swiss$Pop65P,
        swiss$H0OPTOT )
pik=inclusionprobabilities(swiss$POPTOT,400)
sample=balancedstratification(X,swiss$REG,pik,comment=TRUE)
#the sample is
as.character(swiss$Nom[sample==1])
```

**Description**

Selects a balanced two-stage sample.

**Usage**

```
balancedtwostage(X,selection,m,n,PU,comment=TRUE,method=1)
```

**Arguments**

X	matrix of auxiliary variables on which the sample must be balanced.
selection	1, for simple random sampling without replacement at each stage, 2, for self-weighting two-stage selection.
m	number of primary sampling units to be selected.
n	number of second-stage sampling units to be selected.
PU	vector of integers that defines the primary sampling units.
comment	a comment is written during the execution if comment is TRUE.
method	the used method in the function samplecube.

**Value**

The function returns a matrix whose columns are the following five vectors: the selected second-stage sampling units (0 - unselected, 1 - selected), the final inclusion probabilities, the selected primary sampling units, the inclusion probabilities of the first stage, the inclusion probabilities of the second stage.

**See Also**

[samplecube](#), [fastflightcube](#), [landingcube](#), [balancedstratification](#), [balancedcluster](#)

**Examples**

```
#####
## Example 1
#####
# definition of the primary units (3 primary units)
PU=c(1,1,1,1,1,2,2,2,2,3,3,3,3,3)
# matrix of balancing variables
X=cbind(c(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15))
# selection of 2 primary sampling units and 4 second-stage sampling units
s=balancedtwostage(X,1,2,4,PU,comment=TRUE)
# the sample and the inclusion probabilities
s
#####
## Example 2
#####
data(MU284)
X=cbind(MU284$P75,MU284$CS82,MU284$SS82,MU284$ME84)
N=dim(X)[1]
PU=MU284$CL
```

```
m=20
n=60
res=balancedtwostage(X,1,m,n,PU,TRUE)
# the sample and the inclusion probabilities
res
```

---

belgianmunicipalities *The Belgian municipalities population*

---

### Description

This data provides information about the Belgian population of July 1, 2004 compared to that of July 1, 2003, and some financial information about the municipality incomes at the end of 2001.

### Usage

```
data(belgianmunicipalities)
```

### Format

A data frame with 589 observations on the following 17 variables:

**Commune** municipality name.

**INS** 'Institut National de statistique' code.

**Province** province number.

**Arrondiss** administrative division number.

**Men04** number of men on July 1, 2004.

**Women04** number of women on July 1, 2004.

**Tot04** total population on July 1, 2004.

**Men03** number of men on July 1, 2003.

**Women03** number of women on July 1, 2003.

**Tot03** total population on July 1, 2003.

**Diffmen** number of men on July 1, 2004 minus the number of men on July 1, 2003.

**Diffwom** number of women on July 1, 2004 minus the number of women on July 1, 2003.

**DiffTOT** difference between the total population on July 1, 2004 and on July 1, 2003.

**TaxableIncome** total taxable income in euros in 2001.

**Totaltaxation** total taxation in euros in 2001.

**averageincome** average of the income-tax return in euros in 2001.

**medianincome** median of the income-tax return in euros in 2001.

### Source

[http://www.statbel.fgov.be/figures/download\\\_fr.asp](http://www.statbel.fgov.be/figures/download\_fr.asp)

**Examples**

```
data(belgianmunicipalities)
hist(belgianmunicipalities$medianincome)
```

---

calib	<i>g-weights of the calibration estimator</i>
-------	---

---

**Description**

Computes the g-weights of the calibration estimator. The g-weights should lie in the specified bounds for the truncated and logit methods.

**Usage**

```
calib(Xs,d,total,q=rep(1,length(d)),method=c("linear","raking","truncated",
"logit"),bounds=c(low=0,upp=10),description=FALSE,max_iter=500)
```

**Arguments**

Xs	matrix of calibration variables.
d	vector of initial weights.
total	vector of population totals.
q	vector of positive values accounting for heteroscedasticity; the variation of the g-weights is reduced for small values of q.
method	calibration method (linear, raking, logit, truncated).
bounds	vector of bounds for the g-weights used in the truncated and logit methods; 'low' is the smallest value and 'upp' is the largest value.
description	if description=TRUE, summary of initial and final weights are printed, and their boxplots and histograms are drawn; by default, its value is FALSE.
max_iter	maximum number of iterations in the Newton's method.

**Details**

The argument *method* implements the methods given in the paper of Deville and Särndal(1992).

**Value**

Returns the vector of g-weights.

**References**

Cassel, C.-M., Särndal, C.-E., and Wretman, J. (1976). Some results on generalized difference estimation and generalized regression estimation for finite population. *Biometrika*, 63:615–620.

Deville, J.-C. and Särndal, C.-E. (1992). Calibration estimators in survey sampling. *Journal of the American Statistical Association*, 87:376–382.

Deville, J.-C., Särndal, C.-E., and Sautory, O. (1993). Generalized raking procedure in survey sampling. *Journal of the American Statistical Association*, 88:1013–1020.

**See Also**

[checkcalibration](#), [calibev](#), [gencalib](#)

**Examples**

```
#####
## Example 1
#####
# matrix of sample calibration variables
Xs=cbind(
  c(1,1,1,1,1,0,0,0,0,0),
  c(0,0,0,0,0,1,1,1,1,1),
  c(1,2,3,4,5,6,7,8,9,10)
)
# inclusion probabilities
piks=rep(0.2,times=10)
# vector of population totals
total=c(24,26,290)
# the g-weights using the truncated method
g=calib(Xs,d=1/piks,total,method="truncated",bounds=c(0.75,1.2))
# the calibration estimator of X is equal to 'total' vector
tcal=t(g/piks)%*%Xs
# the g-weights are between lower and upper bounds
g
#####
## Example 2
#####
# Example of g-weights (linear, raking, truncated, logit),
# with the data of Belgian municipalities as population.
# Firstly, a sample is selected by means of Poisson sampling.
# Secondly, the g-weights are calculated.
data(belgianmunicipalities)
attach(belgianmunicipalities)
# matrix of calibration variables for the population
X=cbind(
  Men03/mean(Men03),
  Women03/mean(Women03),
  Diffmen,
  Diffwom,
  TaxableIncome/mean(TaxableIncome),
  Totaltaxation/mean(Totaltaxation),
  averageincome/mean(averageincome),
  medianincome/mean(medianincome))
# selection of a sample with expectation size equal to 200
# by means of Poisson sampling
# the inclusion probabilities are proportional to the average income
pik=inclusionprobabilities(averageincome,200)
N=length(pik)          # population size
s=UPpoisson(pik)      # sample
Xs=X[s==1,]          # sample matrix of calibration variables
piks=pik[s==1]       # sample inclusion probabilities
n=length(piks)        # sample size
```

```

# vector of population totals of the calibration variables
total=c(t(rep(1,times=N))%%X)
# the population total
total
# computation of the g-weights
# by means of different calibration methods.
g1=calib(Xs,d=1/piks,total,method="linear")
g2=calib(Xs,d=1/piks,total,method="raking")
g3=calib(Xs,d=1/piks,total,method="truncated",bounds=c(0.5,1.5))
g4=calib(Xs,d=1/piks,total,method="logit",bounds=c(0.5,1.5))
# In some cases, the calibration does not exist
# particularly when bounds are used.
# if the calibration is possible, the calibration estimator of Xs is printed
if(checkcalibration(Xs,d=1/piks,total,g1)$result)
  print(c((g1/piks) %% Xs)) else print("error")
if(!is.null(g2))
  if(checkcalibration(Xs,d=1/piks,total,g2)$result)
    print(c((g2/piks) %% Xs)) else print("error")
if(!is.null(g3))
  if(checkcalibration(Xs,d=1/piks,total,g3)$result)
    print(c((g3/piks) %% Xs)) else print("error")
if(!is.null(g4))
  if(checkcalibration(Xs,d=1/piks,total,g4)$result)
    print(c((g4/piks) %% Xs)) else print("error")
#####
## Example 3
#####
# Example of calibration and adjustment for nonresponse in the 'calibration' vignette
vignette("calibration", package="sampling")

```

---

calibev

*Calibration estimator and its variance estimation*


---

### Description

Computes the calibration estimator of the population total and its variance estimation using the residuals' method.

### Usage

```
calibev(Ys,Xs,total,pikl,d,g,q=rep(1,length(d)),with=FALSE,EPS=1e-6)
```

### Arguments

Ys	vector of interest variable; its size is n, the sample size.
Xs	matrix of sample calibration variables.
total	vector of population totals for calibration.
pikl	matrix of joint inclusion probabilities of the sample units.
d	vector of initial weights of the sample units.

g	vector of g-weights; its size is n, the sample size.
q	vector of positive values accounting for heteroscedasticity; its size is n, the sample size.
with	if TRUE, the variance estimation takes into account the initial weights d; otherwise, the final weights w=g*d are taken into account; by default, its value is FALSE.
EPS	the tolerance in checking the calibration; by default, its value is 1e-6.

### Details

If with is TRUE, the following formula is used

$$\hat{Var}(\hat{Y}_s) = \sum_{k \in s} \sum_{\ell \in s} ((\pi_{k\ell} - \pi_k \pi_\ell) / \pi_{k\ell}) (d_k e_k) (d_\ell e_\ell)$$

else

$$\hat{Var}(\hat{Y}_s) = \sum_{k \in s} \sum_{\ell \in s} ((\pi_{k\ell} - \pi_k \pi_\ell) / \pi_{k\ell}) (w_k e_k) (w_\ell e_\ell)$$

where  $e_k$  denotes the residual of unit k.

### Value

The function returns two values:

cest	the calibration estimator,
evar	its estimated variance.

### References

- Deville, J.-C. and Särndal, C.-E. (1992). Calibration estimators in survey sampling. *Journal of the American Statistical Association*, 87:376–382.
- Deville, J.-C., Särndal, C.-E., and Sautory, O. (1993). Generalized raking procedure in survey sampling. *Journal of the American Statistical Association*, 88:1013–1020.

### See Also

[calib](#)

### Examples

```
#####
## Example
#####
# Example of g-weights (linear, raking, truncated, logit),
# with the data of Belgian municipalities as population.
# Firstly, a sample is selected by means of Poisson sampling.
# Secondly, the g-weights are calculated.
data(belgianmunicipalities)
attach(belgianmunicipalities)
```

```

# matrix of calibration variables for the population
X=cbind(
Men03/mean(Men03),
Women03/mean(Women03),
Diffmen,
Diffwom,
TaxableIncome/mean(TaxableIncome),
Totaltaxation/mean(Totaltaxation),
averageincome/mean(averageincome),
medianincome/mean(medianincome))
# selection of a sample with expected size equal to 200
# by means of Poisson sampling
# the inclusion probabilities are proportional to the average income
pik=inclusionprobabilities(averageincome,200)
N=length(pik)           # population size
s=UPsystematic(pik)    # draws a sample s using systematic sampling
Xs=X[s==1,]           # matrix of sample calibration variables
piks=pik[s==1]         # sample inclusion probabilities
n=length(piks)         # sample size
# vector of population totals of the calibration variables
total=c(t(rep(1,times=N))%*%X)
g1=calib(Xs,d=1/piks,total,method="linear") # computes the g-weights
pikl=UPsystematicpi2(pik) # computes the matrix of the joint inclusion probabilities
pikls=pikl[s==1,s==1]   # the same matrix for the units in s
Ys=Tot04[s==1]         # the variable of interest is Tot04 (for the units in s)
calibev(Ys,Xs,total,pikls,d=1/piks,g1,with=FALSE,EPS=1e-6)

```

---

checkcalibration	<i>Check calibration</i>
------------------	--------------------------

---

## Description

Checks the validity of the calibration. In some cases, the calibration estimators do not exist, and the g-weights do not allow calibration.

## Usage

```
checkcalibration(Xs, d, total, g, EPS=1e-6)
```

## Arguments

Xs	matrix of calibration variables.
d	vector of initial weights.
total	vector of population totals.
g	vector of g-weights.
EPS	the control value used to verify the calibration, by default equal to 1e-6.

**Details**

In the case where calibration is not possible, the 'value' indicates the difference in obtaining the calibration.

**Value**

The function returns the following three objects:

message	a message concerning the calibration,
result	TRUE if the calibration is possible and FALSE, otherwise.
value	value of $\max(\text{abs}(\text{tr}-\text{total})/\text{total})$ , which is used as criterium to validate the calibration, where $\text{tr}=\text{crossprod}(\text{Xs}, \text{g}*\text{d})$ .

**See Also**

[calib](#)

**Examples**

```
# matrix of auxiliary variables
Xs=cbind(c(1,1,1,1,1,0,0,0,0,0),c(0,0,0,0,0,1,1,1,1,1),c(1,2,3,4,5,6,7,8,9,10))
# inclusion probabilities
pik=rep(0.2,times=10)
# vector of totals
total=c(24,26,280)
# the g-weights
g=calib(Xs,d=1/pik,total,method="raking")
# the calibration is possible
checkcalibration(Xs,d=1/pik,total,g)
```

---

cleanstrata

*Clean strata*

---

**Description**

Renumbers a variable of stratification (categorical variable). The strata receive a number from 1 to the last stratum number. The empty strata are suppressed. This function is used in 'balancedstratification'.

**Usage**

```
cleanstrata(strata)
```

**Arguments**

strata	vector of stratum numbers.
--------	----------------------------

**See Also**[balancedstratification](#)**Examples**

```
# definition of the stratification variable
strata=c(-2,3,-2,3,4,4,4,-2,-2,3,4,0,0,0)
# renumber the strata
cleanstrata(strata)
```

cluster

*Cluster sampling***Description**

Cluster sampling with equal/unequal probabilities.

**Usage**

```
cluster(data, clustername, size, method=c("srswor","srswr","poisson",
"systematic"),pik,description=FALSE)
```

**Arguments**

data	data frame or data matrix; its number of rows is N, the population size.
clustername	the name of the clustering variable.
size	sample size.
method	method to select clusters; the following methods are implemented: simple random sampling without replacement (srswor), simple random sampling with replacement (srswr), Poisson sampling (poisson), systematic sampling (systematic); if the method is not specified, by default the method is "srswor".
pik	vector of selection probabilities or auxiliary information used to compute them; this argument is only used for unequal probability sampling (Poisson, systematic). If an auxiliary information is provided, the function uses the <a href="#">inclusion-probabilities</a> function for computing these probabilities. If the method is "srswr" and the sample size is larger than the population size, this vector is normalized to one.
description	a message is printed if its value is TRUE; the message gives the number of selected clusters, the number of units in the population and the number of selected units. By default, the value is FALSE.

**Value**

The function returns a data set with the following information: the selected clusters, the identifier of the units in the selected clusters, the final inclusion probabilities for these units (they are equal for the units included in the same cluster). If method is "srswr", the number of replicates is also given.

**See Also**

[mstage](#), [strata](#), [getdata](#)

**Examples**

```
#####
## Example 1
#####
# Uses the swissmunicipalities data to draw a sample of clusters
data(swissmunicipalities)
# the variable 'REG' has 7 categories in the population
# it is used as clustering variable
# the sample size is 3; the method is simple random sampling without replacement
cl=cluster(swissmunicipalities,clustername=c("REG"),size=3,method="srswor")
# extracts the observed data
# the order of the columns is different from the order in the initial database
getdata(swissmunicipalities, cl)
#####
## Example 2
#####
# the same data as in Example 1
# the sample size is 3; the method is systematic sampling
# the pik vector is randomly generated using the U(0,1) distribution
cl_sys=cluster(swissmunicipalities,clustername=c("REG"),size=3,method="systematic",
pik=runif(7))
# extracts the observed data
getdata(swissmunicipalities,cl_sys)
```

---

disjunctive

*Disjunctive combination*

---

**Description**

Transforms a categorical variable into a matrix of indicators. The values of the categorical variable are integer numbers (positive or negative).

**Usage**

```
disjunctive(strata)
```

**Arguments**

strata            vector of integer numbers.

**See Also**

[balancedstratification](#)

## Examples

```
# definition of the variable of stratification
strata=c(-2,3,-2,3,4,4,4,-2,-2,3,4,0,0,0)
# computation of the matrix
disjunctive(strata)
```

---

fastflightcube

*Fast flight phase for the cube method*

---

## Description

Executes the fast flight phase of the cube method (algorithm of Chauvet and Tillé, 2005, 2006). The data are sorted following the argument order. Inclusion probabilities equal to 0 or 1 are tolerated.

## Usage

```
fastflightcube(X,pik,order=1,comment=TRUE)
```

## Arguments

X	matrix of auxiliary variables on which the sample must be balanced.
pik	vector of inclusion probabilities.
order	1, the data are randomly arranged, 2, no change in data order, 3, the data are sorted in decreasing order.
comment	a comment is written during the execution if comment is TRUE.

## References

Tillé, Y. (2006), *Sampling Algorithms*, Springer.  
Chauvet, G. and Tillé, Y. (2006). A fast algorithm of balanced sampling. *Computational Statistics*, 21/1:53–62.  
Chauvet, G. and Tillé, Y. (2005). New SAS macros for balanced sampling. In INSEE, editor, *Journées de Méthodologie Statistique*, Paris.  
Deville, J.-C. and Tillé, Y. (2004). Efficient balanced sampling: the cube method. *Biometrika*, 91:893–912.  
Deville, J.-C. and Tillé, Y. (2005). Variance approximation under balanced sampling. *Journal of Statistical Planning and Inference*, 128/2:411–425.

## See Also

[samplecube](#)

**Examples**

```
# Matrix of balancing variables
X=cbind(c(1,1,1,1,1,1,1,1,1),c(1,2,3,4,5,6,7,8,9))
# Vector of inclusion probabilities.
# The sample size is 3.
pik=c(1/3,1/3,1/3,1/3,1/3,1/3,1/3,1/3,1/3)
# pikstar is almost a balanced sample and is ready for the landing phase
pikstar=fastflightcube(X,pik,order=1,comment=TRUE)
pikstar
```

gencalib

*g-weights of the generalized calibration estimator***Description**

Computes the g-weights of the generalized calibration estimator. The g-weights should lie in the specified bounds for the truncated and logit methods.

**Usage**

```
gencalib(Xs,Zs,d,total,q=rep(1,length(d)),method=c("linear","raking","truncated","logit"),
bounds=c(low=0,upp=10),description=FALSE,max_iter=500,C=1)
```

**Arguments**

Xs	matrix of calibration variables.
Zs	matrix of instrumental variables with same dimension as Xs.
d	vector of initial weights.
total	vector of population totals.
q	vector of positive values accounting for heteroscedasticity; the variation of the g-weights is reduced for small values of q.
method	calibration method (linear, raking, logit, truncated).
bounds	vector of bounds for the g-weights used in the truncated and logit methods; 'low' is the smallest value and 'upp' is the largest value.
description	if description=TRUE, summary of initial and final weights are printed, and their boxplots and histograms are drawn; by default, its value is FALSE.
max_iter	maximum number of iterations in the Newton's method.
C	value of the centering constant, by default equals 1.

## Details

The generalized calibration or the instrument vector method computes the g-weights  $g_k = F(\lambda'z_k)$ , where  $z_k$  is a vector with values defined for  $k \in s$  (or  $k \in r$  where  $r$  is the set of respondents) and sharing the dimension of the specified auxiliary vector  $x_k$ . The vectors  $z_k$  and  $x_k$  have to be strongly correlated. The vector  $\lambda$  is determined from the calibration equation  $\sum_{k \in s} d_k g_k x_k = \sum_{k \in U} x_k$  or  $\sum_{k \in r} d_k g_k x_k = \sum_{k \in U} x_k$ . The function  $F$  plays the same role as in the calibration method (see [calib](#)). If  $Xs=Zs$  the calibration method is obtain. If the method is "logit" the g-weights will be centered around the constant C, with  $low < C < upp$ . In the calibration method  $C=1$  (see [calib](#)).

## Value

The function returns the vector of g-weights.

## References

- Deville, J.-C. (1998). La correction de la nonréponse par calage ou par échantillonnage équilibré. Paper presented at the *Congrès de l'ACFAS, Sherbrooke, Québec*.
- Deville, J.-C. (2000). Generalized calibration and application for weighting for non-response, *COMPSTAT 2000: proceedings in computational statistics*, p. 65–76.
- Estevao, V.M., and Särndal, C.E. (2000). A functional form approach to calibration. *Journal of Official Statistics*, 16, 379–399.
- Kott, P.S. (2006). Using calibration weighting to adjust for nonresponse and coverage errors. *Survey Methodology*, 32, 133–142.

## See Also

[checkcalibration](#), [calib](#)

## Examples

```
#####
## Example 1
#####
# matrix of sample calibration variables
Xs=cbind(
c(1,1,1,1,1,0,0,0,0,0),
c(0,0,0,0,0,1,1,1,1,1),
c(1,2,3,4,5,6,7,8,9,10))
# inclusion probabilities
piks=rep(0.2,times=10)
# vector of population totals
total=c(24,26,290)
# matrix of instrumental variables
Zs=Xs+matrix(runif(nrow(Xs)*ncol(Xs)),nrow(Xs),ncol(Xs))
# the g-weights using the truncated method
g=gencalib(Xs,Zs,d=1/piks,total,method="truncated",bounds=c(0.5,1.5))
# the calibration estimator of X is equal to the 'total' vector
t(g/piks)%*%Xs
# the g-weights are between lower and upper bounds
```

```

summary(g)
#####
## Example 2
#####
# Example of generalized g-weights (linear, raking, truncated, logit),
# with the data of Belgian municipalities as population.
# Firstly, a sample is selected by means of Poisson sampling.
# Secondly, the g-weights are calculated.
data(belgianmunicipalities)
attach(belgianmunicipalities)
# matrix of calibration variables for the population
X=cbind(Totaltaxation/mean(Totaltaxation),medianincome/mean(medianincome))
# selection of a sample with expected size equal to 200
# by means of Poisson sampling
# the inclusion probabilities are proportional to the average income
pik=inclusionprobabilities(averageincome,200)
N=length(pik)          # population size
s=UPpoisson(pik)       # sample
Xs=X[s==1,]           # sample calibration variable matrix
piks=pik[s==1]        # sample inclusion probabilities
n=length(piks)        # sample size
# vector of population totals of the calibration variables
total=c(t(rep(1,times=N))%*%X)
# the population total
total
Z=cbind(TaxableIncome/mean(TaxableIncome),averageincome/mean(averageincome))
# defines the instrumental variables
Zs=Z[s==1,]
# computation of the generalized g-weights
# by means of different generalized calibration methods
g1=gencalib(Xs,Zs,d=1/piks,total,method="linear")
g2=gencalib(Xs,Zs,d=1/piks,total,method="raking")
g3=gencalib(Xs,Zs,d=1/piks,total,method="truncated",bounds=c(0.5,8))
g4=gencalib(Xs,Zs,d=1/piks,total,method="logit",bounds=c(0.5,1.5))
# In some cases, the calibration does not exist
# particularly when bounds are used.
# if the calibration is possible, the calibration estimator of Xs is printed
if(checkcalibration(Xs,d=1/piks,total,g1)$result) print(c((g1/piks)%*% Xs)) else print("error")
if(!is.null(g2))
if(checkcalibration(Xs,d=1/piks,total,g2)$result) print(c((g2/piks)%*% Xs)) else print("error")
if(!is.null(g3))
if(checkcalibration(Xs,d=1/piks,total,g3)$result) print(c((g3/piks)%*% Xs)) else print("error")
if(!is.null(g4))
if(checkcalibration(Xs,d=1/piks,total,g4)$result) print(c((g4/piks)%*% Xs)) else print("error")
#####
## Example 3
#####
# Example of generalized calibration and adjustment for unit nonresponse in the 'calibration' vignette
vignette("calibration", package="sampling")

```

**Description**

Extracts the observed data from a data frame. The function is used after a sample has been drawn.

**Usage**

```
getdata(data, m)
```

**Arguments**

data	population data frame or data matrix; its number of rows is N, the population size.
m	vector of selected units or sample data frame.

**See Also**

[srswor](#), [UPsystematic](#), [strata](#), [cluster](#), [mstage](#)

**Examples**

```
#####
## Example 1
#####
# Generates artificial data (a 235X3 matrix with 3 columns: state, region, income).
# The variable 'state' has 2 categories (nc and sc);
# the variable 'region' has 3 categories (1, 2 and 3);
# the variable 'income' is generated using the U(0,1) distribution.
data=rbind(matrix(rep("nc",165),165,1,byrow=TRUE),
matrix(rep("sc",70),70,1,byrow=TRUE))
data=cbind.data.frame(data,c(rep(1,100), rep(2,50), rep(3,15), rep(1,30),rep(2,40)),
1000*runif(235))
names(data)=c("state","region","income")
# the inclusion probabilities are computed using the variable 'income'
pik=inclusionprobabilities(data$income,20)
# draw a sample s using systematic sampling (sample size is 20)
s=UPsystematic(pik)
# extracts the observed data
getdata(data,s)
#####
## Example 2
#####
# see other examples in 'strata', 'cluster', 'mstage' help files
```

**Description**

Computes the Hájek estimator of the population total or population mean.

**Usage**

```
Hajekestimator(y,pik,N=NULL,type=c("total","mean"))
```

**Arguments**

y	vector of the variable of interest; its length is equal to n, the sample size.
pik	vector of the first-order inclusion probabilities; its length is equal to n, the sample size.
N	population size; N is only used for the total estimator; for the mean estimator its value is NULL.
type	the estimator type: total or mean.

**See Also**

[HTestimator](#)

**Examples**

```
# Belgian municipalities data base
data(belgianmunicipalities)
# Computes the inclusion probabilities
pik=inclusionprobabilities(belgianmunicipalities$Tot04,200)
N=length(pik)
n=sum(pik)
# Defines the variable of interest
y=belgianmunicipalities$TaxableIncome
# Draws a Poisson sample of expected size 200
s=UPpoisson(pik)
# Computes the Hajek estimator of the population mean
Hajekestimator(y[s==1],pik[s==1],type="mean")
# Computes the Hajek estimator of the population total
Hajekestimator(y[s==1],pik[s==1],N=N,type="total")
```

---

Hajekstrata

*The Hajek estimator for a stratified design*


---

**Description**

Computes the Hájek estimator of the population total or population mean for a stratified design.

**Usage**

```
Hajekstrata(y,pik,strata,N=NULL,type=c("total","mean"),description=FALSE)
```

**Arguments**

y	vector of the variable of interest; its length is equal to n, the sample size.
pik	vector of the first-order inclusion probabilities for the sampled units; its length is equal to n, the sample size.
strata	vector of size n, with elements indicating the unit stratum.
N	vector of population sizes of strata; N is only used for the total estimator; for the mean estimator its value is NULL.
type	the estimator type: total or mean.
description	if TRUE, the estimator is printed for each stratum; by default, FALSE.

**See Also**

[HTstrata](#)

**Examples**

```
# Swiss municipalities data base
data(swissmunicipalities)
# the variable 'REG' has 7 categories in the population
# it is used as stratification variable
# computes the population stratum sizes
table(swissmunicipalities$REG)
# do not run
# 1 2 3 4 5 6 7
# 589 913 321 171 471 186 245
# the sample stratum sizes are given by size=c(30,20,45,15,20,11,44)
# the method is simple random sampling without replacement
# (equal probability, without replacement)
st=strata(swissmunicipalities, stratanames=c("REG"), size=c(30,20,45,15,20,11,44),
method="srswor")
# extracts the observed data
# the order of the columns is different from the order in the swissmunicipalities database
x=getdata(swissmunicipalities, st)
# computes the population sizes of strata
N=table(swissmunicipalities$REG)
N=N[unique(x$REG)]
# computes the Hajek estimator of the variable Pop020
Hajekstrata(x$Pop020,x$Prob,x$Stratum,N,type="total",description=TRUE)
```

---

HTestimator

*The Horvitz-Thompson estimator*


---

**Description**

Computes the Horvitz-Thompson estimator of the population total.

**Usage**

```
HTestimator(y,pik)
```

**Arguments**

**y** vector of the variable of interest; its length is equal to n, the sample size.

**pik** vector of the first-order inclusion probabilities; its length is equal to n, the sample size.

**See Also**

[UPtille](#)

**Examples**

```
# Belgian municipalities data base
data(belgianmunicipalities)
attach(belgianmunicipalities)
# Computes the inclusion probabilities
pik=inclusionprobabilities(Tot04,200)
N=length(pik)
n=sum(pik)
# Defines the variable of interest
y=TaxableIncome
# Draws a Poisson sample of expected size 200
s=UPpoisson(pik)
# Computes the Horvitz-Thompson estimator
HTestimator(y[s==1],pik[s==1])
```

---

HTstrata

*The Horvitz-Thompson estimator for a stratified design*


---

**Description**

Computes the Horvitz-Thompson estimator of the population total for a stratified design.

**Usage**

```
HTstrata(y,pik,strata,description=FALSE)
```

**Arguments**

**y** vector of the variable of interest; its length is equal to n, the sample size.

**pik** vector of the first-order inclusion probabilities for the sampled units; its length is equal to n, the sample size.

**strata** vector of size n, with elements indicating the unit stratum.

**description** if TRUE, the estimator is printed for each stratum; by default, FALSE.

**See Also**[HTestimator](#)**Examples**

```
# Swiss municipalities data base
data(swissmunicipalities)
# the variable 'REG' has 7 categories in the population
# it is used as stratification variable
# computes the population stratum sizes
table(swissmunicipalities$REG)
# do not run
# 1 2 3 4 5 6 7
# 589 913 321 171 471 186 245
# the sample stratum sizes are given by size=c(30,20,45,15,20,11,44)
# the method is simple random sampling without replacement
# (equal probability, without replacement)
st=strata(swissmunicipalities, stratanames=c("REG"), size=c(30,20,45,15,20,11,44),
method="srswor")
# extracts the observed data
# the order of the columns is different from the order in the initial database
x=getdata(swissmunicipalities, st)
# computes the HT estimator of the variable Pop020
HTstrata(x$Pop020, x$Prob, x$Stratum, description=TRUE)
```

---

inclusionprobabilities

*Inclusion probabilities*

---

**Description**

Computes the first-order inclusion probabilities from a vector of positive numbers (for a probability proportional-to-size sampling design).

**Usage**

```
inclusionprobabilities(a,n)
```

**Arguments**

a	vector of positive numbers.
n	sample size.

**See Also**[inclusionprobastrata](#)

**Examples**

```
#####
## Example 1
#####
# a vector of positive numbers
a=1:20
# computation of the inclusion probabilities for a sample size n=12
pik=inclusionprobabilities(a,12)
pik
#####
## Example 2
#####
# Computation of the inclusion probabilities proportional to the number
# of inhabitants in each municipality of the Belgian database.
data(belgianmunicipalities)
pik=inclusionprobabilities(belgianmunicipalities$Tot04,200)
# the first-order inclusion probabilities for each municipality
data.frame(pik=pik,name=belgianmunicipalities$Commune)
# the inclusion probability sum is equal to the sample size
sum(pik)
```

---

inclusionprobastrata    *Inclusion probabilities for a stratified design*

---

**Description**

Computes the inclusion probabilities for a stratified design. The inclusion probabilities are equal in each stratum.

**Usage**

```
inclusionprobastrata(strata,nh)
```

**Arguments**

strata	vector that defines the strata.
nh	vector with the number of units to be selected in each stratum.

**See Also**

[balancedstratification](#)

**Examples**

```
# the strata
strata=c(1,1,1,1,1,2,2,2,2,2,3,3,3,3,3,3)
# sample size in each stratum
nh=c(2,3,3)
inclusionprobastrata(strata,nh)
```

---

 landingcube

*Landing phase for the cube method*


---

### Description

Landing phase of the cube method using linear programming.

### Usage

```
landingcube(X,pikstar,pik,comment=TRUE)
```

### Arguments

X	matrix of auxiliary variables on which the sample must be balanced.
pikstar	vector obtained at the end of the flight phase.
pik	vector of inclusion probabilities.
comment	a comment is written during the execution if comment is TRUE.

### References

Tillé, Y. (2006), *Sampling Algorithms*, Springer.

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

Chauvet, G. and Tillé, Y. (2005). New SAS macros for balanced sampling. In INSEE, editor, *Journées de Méthodologie Statistique*, Paris.

Deville, J.-C. and Tillé, Y. (2004). Efficient balanced sampling: the cube method. *Biometrika*, 91:893–912.

Deville, J.-C. and Tillé, Y. (2005). Variance approximation under balanced sampling. *Journal of Statistical Planning and Inference*, 128/2:411–425.

### See Also

[samplecube](#), [fastflightcube](#)

### Examples

```
# matrix of balancing variables
X=cbind(c(1,1,1,1,1,1,1,1),c(1.1,2.2,3.1,4.2,5.1,6.3,7.1,8.1,9.1))
# Vector of inclusion probabilities
# The sample has the size equal to 3.
pik=c(1/3,1/3,1/3,1/3,1/3,1/3,1/3,1/3,1/3)
# pikstar is almost a balanced sample and is ready for the landing phase
pikstar=fastflightcube(X,pik,order=1,comment=TRUE)
# selection of the sample s
s=landingcube(X,pikstar,pik,comment=TRUE)
round(s)
```

---

mstage	<i>Multistage sampling</i>
--------	----------------------------

---

**Description**

Implements multistage sampling with equal/unequal probabilities.

**Usage**

```
mstage(data, stage=c("stratified","cluster",""), varnames, size,
method=c("srswor","srswr","poisson","systematic"), pik, description=FALSE)
```

**Arguments**

data	data frame or data matrix; its number of rows is N, the population size.
stage	list of sampling type at each stage; the possible values are: "stratified", "cluster" and "" (without stratification or clustering). For multistage element sampling, this argument is not necessary.
varnames	list of stratification or clustering variables.
size	list of sample sizes (in the order in which the samples appear in the multistage sampling).
method	method to select units; the following methods are implemented: simple random sampling without replacement (srswor), simple random sampling with replacement (srswr), Poisson sampling (poisson), systematic sampling (systematic); if the method is not specified, by default, the method is "srswor".
pik	list of selection probabilities or auxiliary information used to compute them; this argument is only used for unequal probability sampling (Poisson, systematic). If an auxiliary information is provided, the function uses the <a href="#">inclusionprobabilities</a> function for computing these probabilities. If the method is "srswr" and the sample size is larger than the population size, the selection probabilities are normalized to one.
description	a message is printed if its value is TRUE; the message gives the number of selected units and the number of the units in the population. By default, its value is FALSE.

**Value**

The function returns a list, which contains the stages (if m is this list, the stage i is m\$'i' etc) and the following information:

ID_unit	the identifier of selected units at each stage.
Prob_ number _stage	the inclusion probability at stage 'number'.
Prob	the final unit inclusion probability given in the last stage; it is the product of the unit inclusion probabilities at each stage.

**See Also**

[cluster](#), [strata](#), [getdata](#)

**Examples**

```
#####
## Example 1
#####
# Two-stage cluster sampling
# Uses the 'swissmunicipalities' data for drawing a sample of units
data(swissmunicipalities)
# the variable 'REG' (region) has 7 categories;
# it is used as clustering variable in the first-stage sample
# the variable 'CT' (canton) has 26 categories;
# it is used as clustering variable in the second-stage sample
# 4 clusters (regions) are selected in the first-stage
# 1 canton is selected in the second-stage from each sampled cluster
# the method is simple random sampling without replacement
# (equal probability, without replacement)
m=mstage(swissmunicipalities,stage=list("cluster","cluster"), varnames=list("REG","CT"),
size=list(4,c(1,1,1,1)), method="srswor")
# the first stage is m$'1' and the second stage is m$'2'
# extracts the observed data
# the order of the columns is different from the order in the swissmunicipalities database
getdata(swissmunicipalities, m)
#####
## Example 2
#####
# Two-stage element sampling
# Example from An and Watts (New SAS procedures for Analysis of Sample Survey Data)
# Generates artificial data (a 235X3 matrix with 3 columns: state, region, income).
# The variable "state" has 2 categories ('nc' and 'sc').
# The variable "region" has 3 categories (1, 2 and 3).
# The variable "income" is generated using the U(0,1) distribution.
data=rbind(matrix(rep("nc",165),165,1,byrow=TRUE),matrix(rep("sc",70),70,1,byrow=TRUE))
data=cbind(data,frame(data,c(rep(1,100),rep(2,50),rep(3,15),rep(1,30),rep(2,40)),
100*runif(235))
names(data)=c("state","region","income")
# the method is simple random sampling with replacement
# 25 units are drawn in the first-stage
# in the second-stage, 10 units are drawn from the already 25 selected units
m=mstage(data,size=list(25,10),method="srswr")
# extracts the observed data
getdata(data,m)
#####
## Example 3
#####
# One-stage stratified cluster sampling
# The same data as in Example 2
# the variable 'state' is used as stratification variable
# 165 units are in the first stratum and 70 in the second one
# the variable 'region' is used as clustering variable
```

```

# 1 cluster (region) is drawn in each stratum using the "srswor" method
m=mstage(data, stage=list("stratified","cluster"), varnames=list("state","region"),
size=list(c(165,70),c(1,1)),method="srswor")
# extracts the observed data
getdata(data,m)
# verify the result using a contingency table
xx=getdata(data,m)
table(xx[[2]]$state,xx[[2]]$region)
#####
## Example 4
#####
# Two-stage cluster sampling
# The same data as in Example 1
data(swissmunicipalities)
# in the first-stage, the clustering variable is 'REG' (region) with 7 categories
# each region is selected with the probability 1/7
# in the second-stage, the clustering variable is 'CT'(canton) with 26 categories
# in the region 1, there are 3 cantons and each canton is selected with the prob. 1/3
# in the region 2, there are 5 cantons and each canton is selected with the prob. 1/5
# in the region 3, there are 3 cantons and each canton is selected with the prob. 1/3
# in the region 4, there is 1 canton, which it is selected with the prob. 1
# in the region 5, there are 7 cantons and each canton is selected with the prob. 1/7
# in the region 6, there are 6 cantons and each canton is selected with the prob. 1/6
# in the region 7, there is 1 canton, which it is selected with the prob. 1
# it is necessary to use a list of selection probabilities at each stage
# prob is the list of the selection probabilities
# the method is systematic sampling (unequal probabilities, without replacement)
# 4 clusters (regions) are drawn in the first-stage
# 1 cluster (canton) is drawn in the second-stage from each selected region
# ls is the list of sizes
ls=list(4,c(1,1,1,1))
prob=list(rep(4/7,7),list(rep(1/3,3),rep(1/5,5),rep(1/3,3),rep(1,1),rep(1/7,7),
rep(1/6,6),rep(1,1)))
m=mstage(swissmunicipalities,stage=list("cluster","cluster"),varnames=list("REG","CT"),
size=ls, method="systematic",pik=prob)
# extracts the observed data
getdata(swissmunicipalities,m)

```

---

 MU284

*The MU284 population*


---

## Description

This data is from Särndal et al (1992), see Appendix B, p. 652.

## Usage

```
data(MU284)
```

**Format**

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

**LABEL** identifier number from 1 to 284.

**P85** 1985 population (in thousands).

**P75** 1975 population (in thousands).

**RMT85** revenues from 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).

**Source**

<http://lib.stat.cmu.edu/datasets/mu284>

**References**

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

**Examples**

```
data(MU284)
hist(MU284$RMT85)
```

---

postest

*The poststratified estimator*

---

**Description**

Computes the poststratified estimator of the population total.

**Usage**

```
postest(data, y, pik, NG, description=FALSE)
```

**Arguments**

data	data frame or data matrix; its number of rows is n, the sample size.
y	vector of the variable of interest; its length is equal to n, the sample size.
pik	vector of the first-order inclusion probabilities for the sampled units; its length is equal to n, the sample size.
NG	vector of population frequency in each group G; for stratified sampling with poststratification, NG is a matrix of population frequency in each cell GH.
description	if TRUE, the estimator is printed for each poststratum; by default, FALSE.

**See Also**

[poststrata](#)

**Examples**

```
#####
## Example 1
#####
#stratified sampling and poststratification
# Swiss municipalities data base
data(swissmunicipalities)
attach(swissmunicipalities)
# the variable 'REG' has 7 categories in the population
# it is used as stratification variable
# Computes the population stratum sizes
table(swissmunicipalities$REG)
# do not run
# 1 2 3 4 5 6 7
# 589 913 321 171 471 186 245
# the sample stratum sizes are given by size=c(30,20,45,15,20,11,44)
# the method is simple random sampling without replacement
st=strata(swissmunicipalities, stratanames=c("REG"),
size=c(30,20,45,15,20,11,44), method="srswor")
# extracts the observed data
# the order of the columns is different from the order in the initial database
x=getdata(swissmunicipalities, st)
px=poststrata(x,"REG")
ct=unique(px$data$REG)
yy=numeric(length(ct))
for(i in 1:length(ct))
  {xx=swissmunicipalities[REG==ct[i],]
  yy[i]=nrow(xx)
  }
yy
postest(px$data,y=px$data$Pop020,pik=px$data$Prob,NG=diag(yy),description=TRUE)
HTstrata(x$Pop020,x$Prob,x$Stratum)
#the two estimators are equal
#####
## Example 2
#####
```

```

# systematic sampling and poststratification
# Belgian municipalities data base
data(belgianmunicipalities)
Tot=belgianmunicipalities$Tot04
name=belgianmunicipalities$Commune
pik=inclusionprobabilities(Tot,200)
#selects a sample
s=UPsystematic(pik)
#the sample is
as.vector(name[s==1])
# extracts the observed data
b=getdata(belgianmunicipalities,s)
attach(belgianmunicipalities)
pb=poststrata(b,"Province")
#computes the population frequency in each group
ct=unique(pb$data$Province)
yy=numeric(length(ct))
for(i in 1:length(ct))
  {xx=belgianmunicipalities[Province==ct[i],]
  yy[i]=nrow(xx)
  }
postest(pb$data,y=pb$data$TaxableIncome,pik=pik[s==1],NG=yy,description=TRUE)
HTestimator(pb$data$TaxableIncome,pik=pik[s==1])
#####
## Example 3
#####
#cluster sampling and postratification
# Swiss municipalities data base
data(swissmunicipalities)
# the variable 'REG' has 7 categories in the population
# it is used as clustering variable
# the sample size is 3; the method is simple random sampling without replacement
cl=cluster(swissmunicipalities,clustername=c("REG"),size=3,method="srswor")
# extracts the observed data
# the order of the columns is different from the order in the initial database
c=getdata(swissmunicipalities, cl)
pc=poststrata(c,"CT")
#computes the population frequency in each group
ct=unique(pc$data$CT)
yy=numeric(length(ct))
for(i in 1:length(ct))
  {xx=swissmunicipalities[CT==ct[i],]
  yy[i]=nrow(xx)
  }
postest(pc$data,y=pc$data$Pop020,pik=pc$data$Prob,NG=yy,description=TRUE)
#####
## Example 4
#####
#postratification with two criteria
#artificial data frame
data=rbind(matrix(rep("nc",165),165,1,byrow=TRUE),matrix(rep("sc",70),70,1,byrow=TRUE))
data=cbind.data.frame(data,c(rep(1,100), rep(2,50), rep(3,15), rep(1,30),rep(2,40)),
1000*runif(235))

```

```

names(data)=c("state","region","income")
# computes the population stratum sizes
table(data$region,data$state)
# not run
#   nc  sc
#  1 100  30
#  2  50  40
#  3  15   0
#selects a sample of size 10
s=srswor(10,nrow(data))
# poststratification using region and state
ps=poststrata(data[s==1,],c("region","state"))
#computes the population frequency in each group
ct=unique(ps$data$poststratum)
yy=numeric(length(ct))
for(i in 1:length(ct))
{
  xy=ps$data[ps$data$poststratum==ct[i],]
  xstate=unique(xy$state)
  ystate=unique(xy$region)
  xx=data[data$state==xstate & data$region==ystate,]
  yy[i]=nrow(xx)
}
postest(ps$data,y=ps$data$income,pik=rep(10/nrow(data),10),NG=yy,description=TRUE)

```

---

poststrata

*Poststratification*


---

### Description

Poststratification using several criteria.

### Usage

```
poststrata(data, postnames = NULL)
```

### Arguments

data	data frame or data matrix; its number of rows is n, the sample size.
postnames	vector of poststratification variables.

### Value

The function	produces an object, which contains the following information:
data	the final data frame with a new column ('poststratum') containing the unit poststratum.
npost	the number of poststrata.

**See Also**[postest](#)**Examples**

```

# Example from An and Watts (New SAS procedures for Analysis of Sample Survey Data)
# generates artificial data (a 235X3 matrix with 3 columns: state, region, income).
# the variable "state" has 2 categories ('nc' and 'sc').
# the variable "region" has 3 categories (1, 2 and 3).
# the income variable is randomly generated
data=rbind(matrix(rep("nc",165),165,1,byrow=TRUE),matrix(rep("sc",70),70,1,byrow=TRUE))
data=cbind.data.frame(data,c(rep(1,100), rep(2,50), rep(3,15), rep(1,30),rep(2,40)),
1000*runif(235))
names(data)=c("state","region","income")
# computes the population stratum sizes
table(data$region,data$state)
# not run
#   nc  sc
# 1 100 30
# 2  50 40
# 3  15  0
# postratification using two criteria: state and region
poststrata(data,postnames=c("state","region"))

```

---

**ratioest***The ratio estimator*

---

**Description**

Computes the ratio estimator of the population total.

**Usage**

```
ratioest(y,x,Tx,pik)
```

**Arguments**

y	vector of the variable of interest; its length is equal to n, the sample size.
x	vector of auxiliary information; its length is equal to n, the sample size.
Tx	population total of x.
pik	vector of the first-order inclusion probabilities; its length is equal to n, the sample size.

**Value**

The function returns the value of the ratio estimator.

**See Also**[regest](#)**Examples**

```

data(MU284)
# there are 3 outliers which are deleted from the population
MU281=MU284[MU284$RMT85<=3000,]
attach(MU281)
# computes the inclusion probabilities using the variable P85; sample size 120
pik=inclusionprobabilities(P85,120)
# defines the variable of interest
y=RMT85
# defines the auxiliary information
x=CS82
# draws a systematic sample of size 120
s=UPsystematic(pik)
# computes the ratio estimator
ratioest(y[s==1],x[s==1],sum(x),pik[s==1])

```

ratioest\_strata

*The ratio estimator for a stratified design***Description**

Computes the ratio estimator of the population total for a stratified design. The ratio estimator of a total is the sum of ratio estimator in each stratum.

**Usage**

```
ratioest_strata(y,x,TX_strata,pik,strata,description=FALSE)
```

**Arguments**

<code>y</code>	vector of the variable of interest; its length is equal to <code>n</code> , the sample size.
<code>x</code>	vector of auxiliary information; its length is equal to <code>n</code> , the sample size.
<code>TX_strata</code>	vector of population x-total in each stratum; its length is equal to the number of strata.
<code>pik</code>	vector of the first-order inclusion probabilities; its length is equal to <code>n</code> , the sample size.
<code>strata</code>	vector of size <code>n</code> , with elements indicating the unit stratum.
<code>description</code>	if TRUE, the ratio estimator in each stratum is printed; by default, it is FALSE.

**Value**

The function returns the value of the ratio estimator.

**See Also**[ratioest](#)**Examples**

```
#####
# Example 1
#####
# this example uses MU284 data with the 'REG' variable for stratification
data(MU284)
attach(MU284)
# there are 3 outliers which are deleted from the population
MU281=MU284[RMT85<=3000,]
detach(MU284)
attach(MU281)
# computes the inclusion probabilities using the variable P85
pik=inclusionprobabilities(P85,120)
# defines the variable of interest
y=RMT85
# defines the auxiliary information
x=CS82
# computes the population stratum sizes
table(MU281$REG)
# not run
# 1 2 3 4 5 6 7 8
# 24 48 32 37 55 41 15 29
# a sample is drawn in each region
# the sample stratum sizes are given by size=c(4,10,8,4,6,4,6,7)
s=strata(MU281,c("REG"),size=c(4,10,8,4,6,4,6,7), method="systematic",pik=P85)
# extracts the observed data
MU281sample=getdata(MU281,s)
# computes the population x-totals in each stratum
TX_strata=as.vector(tapply(CS82,list(REG),FUN=sum))
# computes the ratio estimator
ratioest_strata(MU281sample$RMT85,MU281sample$CS82,TX_strata,
MU281sample$Prob,MU281sample$Stratum)
#####
# Example 2
#####
# this is an artificial example (see Example 1 in the 'strata' function)
# there are 4 columns: state, region, income and aux
# 'income' is the variable of interest, and 'aux' is the auxiliary information
# which is correlated to the income
data=rbind(matrix(rep("nc",165),165,1,byrow=TRUE),matrix(rep("sc",70),70,1,byrow=TRUE))
data=cbind.data.frame(data,c(rep(1,100), rep(2,50), rep(3,15), rep(1,30),rep(2,40)),
1000*runif(235))
names(data)=c("state","region","income")
attach(data)
aux=income+rnorm(length(income),0,1)
data=cbind.data.frame(data,aux)
# computes the population stratum sizes
table(data$region,data$state)
```

```

# not run
#   nc  sc
#  1 100 30
#  2  50 40
#  3  15  0
# there are 5 cells with non-zero values; one draws 5 samples (1 sample in each stratum)
# the sample stratum sizes are 10,5,10,4,6, respectively
# the method is 'srswor' (equal probability, without replacement)
s=strata(data,c("region","state"),size=c(10,5,10,4,6), method="srswor")
# extracts the observed data
xx=getdata(data,s)
# computes the population x-total for each stratum
TX_strata=na.omit(as.vector(tapply(aux,list(region,state),FUN=sum)))
# computes the ratio estimator
ratioest_strata(xx$income,xx$aux,TX_strata,xx$Prob,xx$Stratum,description=TRUE)

```

---

regest

*The regression estimator*


---

### Description

Computes the regression estimator of the population total, using the design-based approach.

### Usage

```
regest(formula,Tx,weights,pikl,n,sigma=rep(1,length(weights)))
```

### Arguments

formula	the regression model formula ( $y \sim x$ ).
Tx	population total of x, the auxiliary variable.
weights	vector of the weights; its length is equal to n, the sample size.
pikl	the matrix of joint inclusion probabilities for the sample.
n	the sample size.
sigma	vector of positive values accounting for heteroscedasticity.

### Value

The function returns a list containing the following components:

regest	the value of the regression estimator.
coefficients	a vector of beta coefficients.
std_error	the standard error of coefficients.
t_value	the t-values associated to the coefficients.
p_value	the p-values associated to the coefficients.
cov_mat	the covariance matrix of the coefficients.

weights           the specified weights.  
 y                 the response variable.  
 x                 the model matrix.

### See Also

[ratioest](#), [regest\\_strata](#)

### Examples

```
# uses the MU284 population to draw a systematic sample
data(MU284)
# there are 3 outliers which are deleted from the population
MU281=MU284[MU284$RMT85<=3000,]
attach(MU281)
# computes the inclusion probabilities using the variable P85; sample size 40
pik=inclusionprobabilities(P85,40)
# the joint inclusion probabilities for systematic sampling
pikl=UPsystematicpi2(pik)
# draws a systematic sample of size 40
s=UPsystematic(pik)
# defines the variable of interest
y=RMT85[s==1]
# defines the auxiliary information
x1=CS82[s==1]
x2=SS82[s==1]
# the joint inclusion probabilities for s
pikls=pikl[s==1,s==1]
# the first-order inclusion probabilities for s
piks=pik[s==1]
# computes the regression estimator with the model  $y \sim x_1 + x_2 - 1$ 
r=regest(formula=y~x1+x2-1,Tx=c(sum(CS82),sum(SS82)),weights=1/piks,pikl=pikls,n=40)
# the regression estimator is
r$regest
# the beta coefficients are
r$coefficients
# regression estimator is the same as the calibration estimator
Xs=cbind(x1,x2)
total=c(sum(CS82),sum(SS82))
g1=calib(Xs,d=1/piks,total,method="linear")
checkcalibration(Xs,d=1/piks,total,g1)
calibev(y,Xs,total,pikls,d=1/piks,g1,with=TRUE,EPS=1e-6)
```

---

regest\_strata

*The regression estimator for a stratified design*

---

### Description

Computes the regression estimator of the population total, using the design-based approach, for a stratified sampling. The same regression model is used for all strata.

**Usage**

```
regest_strata(formula, weights, Tx_strata, strata, pikl,
              sigma=rep(1, length(weights)), description=FALSE)
```

**Arguments**

formula	the regression model formula ( $y \sim x$ ).
weights	vector of the weights; its length is equal to $n$ , the sample size.
Tx_strata	population total of $x$ , the auxiliary variable.
strata	vector of stratum identifier.
pikl	the joint inclusion probabilities for the sample.
sigma	vector of positive values accounting for heteroscedasticity.
description	if TRUE, the following components are printed for each stratum: the Horvitz-Thompson estimator, the beta coefficients, their standard error, $t\_values$ , $p\_values$ , and the covariance matrix. By default, FALSE.

**Value**

The function returns the value of the regression estimator computed as the sum of the stratum estimators.

**See Also**

[regest](#)

**Examples**

```
# generates artificial data
y=rgamma(10,3)
x=y+rnorm(10)
Stratum=c(1,1,2,2,2,3,3,3,3,3)
# population size
N=200
# sample size
n=10
# assume proportional allocation, nh/Nh=n/N
pikl=matrix(0,n,n)
for(i in 1:n)
  {for(j in 1:n)
    if(i!=j)
      pikl[i,j]=pikl[j,i]=n*(n-1)/(N*(N-1))
    pikl[i,i]=n/N
  }
regest_strata(formula=y~x-1, weights=rep(N/n,n), Tx_strata=c(50,30,40),
              strata=Stratum, pikl, description=TRUE)
```

---

rhg	<i>Response homogeneity groups</i>
-----	------------------------------------

---

**Description**

Computes the response homogeneity groups and the response probability for each unit in these groups.

**Usage**

```
rhg(X,selection)
```

**Arguments**

X	sample data frame; it should contain the columns 'ID_unit' and 'status'; 'ID_unit' denotes the unit identifier (a number); 'status' is a 1/0 variable denoting the response/non-response of a unit.
selection	vector of variable names in X used to construct the groups.

**Details**

Into a response homogeneity group, the response probability is the same for all units. Data are missing at random within groups, conditionally on the selected sample.

**Value**

The initial sample data frame and also the following components:

rhgroup	the response homogeneity group for each unit.
prob_response	the response probability for each unit; for the units with status=0, this probability is 0.

**References**

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

**See Also**

[rhg\\_strata](#), [calib](#)

**Examples**

```
# defines the inclusion probabilities for the population
pik=c(0.2,0.7,0.8,0.5,0.4,0.4)
# X is the population data frame
X=cbind.data.frame(pik,c("A","B","A","A","C","B"))
names(X)=c("Prob","town")
# selects a sample using systematic sampling
```

```

s=UPsystematic(pik)
# Xs is the sample data frame
Xs=getdata(X,s)
# adds the status column to Xs (1 - sample respondent, 0 otherwise)
Xs=cbind.data.frame(Xs,status=c(1,0,1))
# creates the response homogeneity groups using the 'town' variable
rhg(Xs,selection="town")

```

---

rhg\_strata

*Response homogeneity groups for a stratified sampling*


---

### Description

Computes the response homogeneity groups and the response probability for each unit in these groups for a stratified sampling.

### Usage

```
rhg_strata(X,selection)
```

### Arguments

X	sample data frame; it should contain the columns 'ID_unit', 'Stratum', and 'status'; 'ID_unit' denotes the unit identifier (a number); 'Stratum' denotes the unit stratum; 'status' is a 1/0 variable denoting the response/non-response of a unit in the sample.
selection	vector of variable names in X used to construct the groups.

### Details

Into a response homogeneity group, the response probability is the same for all units. Data are missing at random within groups, conditionally on the selected sample.

### Value

The initial sample data frame and also the following components:

rhgroup	the response homogeneity group for each unit conditionally on its stratum.
prob_response	the response probability for each unit; for the units with status=0, this probability is 0.

### References

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

### See Also

[rhg](#), [calib](#)

## Examples

```
#####
## Example 1
#####
# uses Example 2 from the 'strata' function help file
data=rbind(matrix(rep("nc",165),165,1,byrow=TRUE),matrix(rep("sc",70),70,1,byrow=TRUE))
data=cbind.data.frame(data,c(rep(1,100), rep(2,50), rep(3,15), rep(1,30),rep(2,40)),
1000*runif(235))
names(data)=c("state","region","income")
# draws a sample
s1=strata(data,c("region","state"),size=c(10,5,10,4,6), method="systematic",
pik=data$income)
# extracts the observed data
s1=getdata(data,s1)
# generates randomly the 'status' variable (1-sample respondent, 0-otherwise)
status=runif(nrow(s1))
for(i in 1:length(status))
  if(status[i]<0.3) status[i]=0 else status[i]=1
# adds the 'status' variable to the sample data frame s1
s1=cbind.data.frame(s1,status)
# creates classes of income using the median of income
# suppose that the income is available for all units in sample
classincome=numeric(nrow(s1))
for(i in 1:length(classincome))
  if(s1$income[i]<median(s1$income)) classincome[i]=1 else classincome[i]=2
# adds 'classincome' to s1
s1=cbind.data.frame(s1,classincome)
# computes the response homogeneity groups using the 'classincome' variable
rhg_strata(s1,selection=c("classincome"))
#####
## Example 2
#####
# the same data as in Example 1
# but we also add the 'sex' column (1-female, 2-male)
# suppose that the sex is available for all units in sample
sex=c(rep(1,12),rep(2,8),rep(1,10),rep(2,5))
s1=cbind.data.frame(s1,sex)
# computes the response homogeneity groups using the 'classincome' and 'sex' variables
rhg_strata(s1,selection=c("classincome","sex"))
```

---

rmodel

*Response probability using logistic regression*

---

## Description

Computes the response probabilities using logistic regression for non-response adjustment. For stratified sampling, the same logistic model is used for all strata.

## Usage

```
rmodel(formula,weights,X)
```

**Arguments**

formula	the regression model formula ( $y \sim x$ ).
weights	vector of the weights; its length is equal to n, the sample size.
X	the sample data frame.

**Value**

The function returns the sample data frame with a new column 'prob\_resp', which contains the response probabilities.

**See Also**

[rhg](#)

**Examples**

```
# Example from An and Watts (New SAS procedures for Analysis of Sample Survey Data)
# generates artificial data (a 235X3 matrix with 3 columns: state, region, income).
# the variable "state" has 2 categories ('nc' and 'sc').
# the variable "region" has 3 categories (1, 2 and 3).
# the sampling frame is stratified by region within state.
# the income variable is randomly generated
data=rbind(matrix(rep("nc",165),165,1,byrow=TRUE),matrix(rep("sc",70),70,1,byrow=TRUE))
data=cbind.data.frame(data,c(rep(1,100), rep(2,50), rep(3,15), rep(1,30),rep(2,40)),
1000*runif(235))
names(data)=c("state","region","income")
# computes the population stratum sizes
table(data$region,data$state)
# not run
#   nc  sc
# 1 100 30
# 2  50 40
# 3  15  0
# there are 5 cells with non-zero values; one draws 5 samples (1 sample in each stratum)
# the sample stratum sizes are 10,5,10,4,6, respectively
# the method is 'srswor' (equal probability, without replacement)
s=strata(data,c("region","state"),size=c(10,5,10,4,6), method="srswor")
# extracts the observed data
x=getdata(data,s)
# generates randomly the 'status' column (1 - respondent, 0 - nonrespondent)
status=round(runif(nrow(x)))
x=cbind(x,status)
# computes the response probabilities
rmodel(x$status~x$income+x$Stratum,weights=1/x$Prob,x)
# the same example without stratification
rmodel(x$status~x$income,weights=1/x$Prob,x)
```

samplecube

*Sample cube method***Description**

Selects a balanced sample (a vector of 0 and 1) or an almost balanced sample. Firstly, the flight phase is applied. Next, if needed, the landing phase is applied on the result of the flight phase.

**Usage**

```
samplecube(X,pik,order=1,comment=TRUE,method=1)
```

**Arguments**

X	matrix of auxiliary variables on which the sample must be balanced.
pik	vector of inclusion probabilities.
order	1, the data are randomly arranged, 2, no change in data order, 3, the data are sorted in decreasing order.
comment	a comment is written during the execution if comment is TRUE.
method	1, for a landing phase by linear programming, 2, for a landing phase by suppression of variables.

**References**

Tillé, Y. (2006), *Sampling Algorithms*, Springer.  
 Chauvet, G. and Tillé, Y. (2006). A fast algorithm of balanced sampling. *Computational Statistics*, 21/1:53–62.  
 Chauvet, G. and Tillé, Y. (2005). New SAS macros for balanced sampling. In INSEE, editor, *Journées de Méthodologie Statistique*, Paris.  
 Deville, J.-C. and Tillé, Y. (2004). Efficient balanced sampling: the cube method. *Biometrika*, 91:893–912.  
 Deville, J.-C. and Tillé, Y. (2005). Variance approximation under balanced sampling. *Journal of Statistical Planning and Inference*, 128/2:411–425.

**See Also**

[landingcube](#), [fastflightcube](#)

**Examples**

```
#####
## Example 1
#####
# matrix of balancing variables
X=cbind(c(1,1,1,1,1,1,1,1,1),c(1.1,2.2,3.1,4.2,5.1,6.3,7.1,8.1,9.1))
# vector of inclusion probabilities
```

```

# the sample size is 3.
pik=c(1/3,1/3,1/3,1/3,1/3,1/3,1/3,1/3,1/3)
# selection of the sample
s=samplecube(X,pik,order=1,comment=TRUE)
# The selected sample
(1:length(pik))[s==1]
#####
## Example 2
#####
# 2 strata and 2 auxiliary variables
# we verify the values of the inclusion probabilities by simulations
X=rbind(c(1,0,1,2),c(1,0,2,5),c(1,0,3,7),c(1,0,4,9),
c(1,0,5,1),c(1,0,6,5),c(1,0,7,7),c(1,0,8,6),c(1,0,9,9),
c(1,0,10,3),c(0,1,11,3),c(0,1,12,2),c(0,1,13,3),
c(0,1,14,6),c(0,1,15,8),c(0,1,16,9),c(0,1,17,1),
c(0,1,18,2),c(0,1,19,3),c(0,1,20,4))
pik=rep(1/2,times=20)
ppp=rep(0,times=20)
sim=100 #for accurate results increase this value
for(i in (1:sim))
ppp=ppp+samplecube(X,pik,1,FALSE)
ppp=ppp/sim
print(ppp)
print(pik)
#####
## Example 3
#####
# unequal probability sampling by cube method
# one auxiliary variable equal to the inclusion probability
N=200
pik=runif(N)
pikfin=samplecube(array(pik,c(N,1)),pik,1,TRUE)
#####
## Example 4
#####
# p auxiliary variables generated randomly
N=1000
p=7
x=rnorm(N*p,10,3)
# random inclusion probabilities
pik= runif(N)
X=array(x,c(N,p))
X=cbind(cbind(X,rep(1,times=N)),pik)
pikfin=samplecube(X,pik,1,TRUE)
#####
## Example 5
#####
# strata and an auxiliary variable
N=5000
a=rep(1,times=N)
b=rep(0,times=N)
V1=c(a,b,b)
V2=c(b,a,b)

```

```

V3=c(b,b,a)
X=cbind(V1,V2,V3)
pik=rep(2/10,times=3*N)
pikfin=samplecube(X,pik,1,TRUE)
#####
## Example 6
#####
# Selection of a balanced sample using the MU284 population,
# simulation and comparison of the variance with
# unequal probability sampling of fixed sample size.
#####
data(MU284)
# Computation of the inclusion probabilities
pik=inclusionprobabilities(MU284$P75,50)
# Definition of the matrix of balancing variables
X=cbind(MU284$P75,MU284$CS82,MU284$SS82,MU284$S82,MU284$ME84,MU284$REV84)
# Computation of the Horvitz-Thompson estimator for a balanced sample
s=samplecube(X,pik,1,FALSE)
HTestimator(MU284$RMT85[s==1],pik[s==1])
# Computation of the Horvitz-Thompson estimator for an unequal probability sample
s=samplecube(matrix(pik),pik,1,FALSE)
HTestimator(MU284$RMT85[s==1],pik[s==1])
# simulations; for a better accuracy, increase the value of 'sim'
sim=8
res1=rep(0,times=sim)
res2=rep(0,times=sim)
for(i in 1:sim)
{
cat("Simulation number ",i,"\n")
s=samplecube(X,pik,1,FALSE)
res1[i]=HTestimator(MU284$RMT85[s==1],pik[s==1])
s=samplecube(matrix(pik),pik,1,FALSE)
res2[i]=HTestimator(MU284$RMT85[s==1],pik[s==1])
}
# summary and boxplots
summary(res1)
summary(res2)
ss=cbind(res1,res2)
colnames(ss) = c("balanced sampling","uneq prob sampling")
boxplot(data.frame(ss), las=1)

```

## Description

Draws a simple random sampling without replacement of size  $n$  (equal probabilities, fixed sample size, without replacement).

**Usage**

```
srswor(n,N)
```

**Arguments**

```
n          sample size.
N          population size.
```

**Value**

Returns a vector (with elements 0 and 1) of size N, the population size. Each element k of this vector indicates the status of unit k (1, unit k is selected in the sample; 0, otherwise).

**See Also**

[sample](#), [srswr](#)

**Examples**

```
#####
## Example 1
#####
#select a sample
s=srswor(3,10)
#the sample is
(1:10)[s==1]
#####
## Example 2
#####
data(belgianmunicipalities)
Tot=belgianmunicipalities$Tot04
name=belgianmunicipalities$Commune
n=200
#select a sample
s=srswor(n,length(Tot))
#the sample is
as.vector(name[s==1])
```

---

```
srswor1
```

```
Selection-rejection method
```

---

**Description**

Draws a simple random sampling without replacement of size n using the selection-rejection method.

**Usage**

```
srswor1(n,N)
```

**Arguments**

n                    sample size.  
N                    population size.

**Value**

Returns a vector (with elements 0 and 1) of size N, the population size. Each element k of this vector indicates the status of unit k (1, unit k is selected in the sample; 0, otherwise).

**References**

Fan, C.T., Muller, M.E., Rezucha, I. (1962), Development of sampling plans by using sequential (item by item) selection techniques and digital computer, *Journal of the American Statistical Association*, 57, 387-402.

**See Also**

[srswor](#)

**Examples**

```
s=srswor1(3,10)
#the sample is
(1:10)[s==1]
```

---

srswr

*Simple random sampling with replacement*

---

**Description**

Draws a simple random sampling with replacement of size n (equal probabilities, fixed sample size, with replacement).

**Usage**

```
srswr(n,N)
```

**Arguments**

n                    sample size.  
N                    population size.

**Value**

Returns a vector of size N, population size. Each element k of this vector indicates the number of replicates for unit k in the sample.

**See Also**

[rmultinom](#), [UPmultinomial](#)

**Examples**

```
s=srswr(3,10)
#the selected units are
(1:10)[s!=0]
#with the number of replicates
s[s!=0]
```

---

strata	<i>Stratified sampling</i>
--------	----------------------------

---

**Description**

Stratified sampling with equal/unequal probabilities.

**Usage**

```
strata(data, stratanames=NULL, size, method=c("srswor", "srswr", "poisson",
"systematic"), pik, description=FALSE)
```

**Arguments**

data	data frame or data matrix; its number of rows is N, the population size.
stratanames	vector of stratification variables.
size	vector of stratum sample sizes (in the order in which the strata are given in the input data set).
method	method to select units; the following methods are implemented: simple random sampling without replacement (srswor), simple random sampling with replacement (srswr), Poisson sampling (poisson), systematic sampling (systematic); if "method" is missing, the default method is "srswor".
pik	vector of inclusion probabilities or auxiliary information used to compute them; this argument is only used for unequal probability sampling (Poisson and systematic). If an auxiliary information is provided, the function uses the <a href="#">inclusion-probabilities</a> function for computing these probabilities. If the method is "srswr" and the sample size is larger than the population size, this vector is normalized to one.
description	a message is printed if its value is TRUE; the message gives the number of selected units and the number of the units in the population. By default, the value is FALSE.

**Details**

The data should be sorted in ascending order by the columns given in the stratanames argument before applying the function. Use, for example, data[order(data\$state,data\$region),].

**Value**

The function produces an object, which contains the following information:

ID_unit	the identifier of the selected units.
Stratum	the unit stratum.
Prob	the final unit inclusion probability.

**See Also**

[getdata](#), [mstage](#)

**Examples**

```
#####
## Example 1
#####
# Example from An and Watts (New SAS procedures for Analysis of Sample Survey Data)
# generates artificial data (a 235X3 matrix with 3 columns: state, region, income).
# the variable "state" has 2 categories ('nc' and 'sc').
# the variable "region" has 3 categories (1, 2 and 3).
# the sampling frame is stratified by region within state.
# the income variable is randomly generated
data=rbind(matrix(rep("nc",165),165,1,byrow=TRUE),matrix(rep("sc",70),70,1,byrow=TRUE))
data=cbind.data.frame(data,c(rep(1,100), rep(2,50), rep(3,15), rep(1,30),rep(2,40)),
1000*runif(235))
names(data)=c("state","region","income")
# computes the population stratum sizes
table(data$region,data$state)
# not run
#   nc  sc
#  1 100  30
#  2  50  40
#  3  15   0
# there are 5 cells with non-zero values
# one draws 5 samples (1 sample in each stratum)
# the sample stratum sizes are 10,5,10,4,6, respectively
# the method is 'srswor' (equal probability, without replacement)
s=strata(data,c("region","state"),size=c(10,5,10,4,6), method="srswor")
# extracts the observed data
getdata(data,s)
# see the result using a contingency table
table(s$region,s$state)
#####
## Example 2
#####
# The same data as in Example 1
# the method is 'systematic' (unequal probability, without replacement)
# the selection probabilities are computed using the variable 'income'
s=strata(data,c("region","state"),size=c(10,5,10,4,6), method="systematic",pik=data$income)
# extracts the observed data
getdata(data,s)
```

```

# see the result using a contingency table
table(s$region,s$state)
#####
## Example 3
#####
# Uses the 'swissmunicipalities' data as population for drawing a sample of units
data(swissmunicipalities)
# the variable 'REG' has 7 categories in the population
# it is used as stratification variable
# Computes the population stratum sizes
table(swissmunicipalities$REG)
# do not run
# 1 2 3 4 5 6 7
# 589 913 321 171 471 186 245
# the sample stratum sizes are given by size=c(30,20,45,15,20,11,44)
# 30 units are drawn from the first stratum, 20 in the second one, etc.
# the method is simple random sampling without replacement
# (equal probability, without replacement)
st=strata(swissmunicipalities,stratanames=c("REG"),size=c(30,20,45,15,20,11,44),
method="srswor")
# extracts the observed data
# the order of the columns is different from the order in the initial database
getdata(swissmunicipalities, st)
# see the result using a contingency table
table(st$REG)

```

---

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)
```

---

UPbrewer

*Brewer sampling*

---

### Description

Uses the Brewer's method to select a sample of units (unequal probabilities, without replacement, fixed sample size).

### Usage

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

### Arguments

pik	vector of the inclusion probabilities.
eps	the control value, by default equal to 1e-06; it is used to control pik (pik>eps & pik < 1-eps).

**Value**

Returns a vector (with elements 0 and 1) of size N, the population size. Each element k of this vector indicates the status of unit k (1, unit k is selected in the sample; 0, otherwise).

**References**

Brewer, K. (1975), A simple procedure for  $\pi$ pswor, *Australian Journal of Statistics*, 17:166-172.

**See Also**

[UPsystematic](#)

**Examples**

```
#define the inclusion probabilities
pik=c(0.2,0.7,0.8,0.5,0.4,0.4)
#select a sample
s=UPbrewer(pik)
#the sample is
(1:length(pik))[s==1]
```

---

UPmaxentropy

*Maximum entropy sampling with fixed sample size and unequal probabilities*


---

**Description**

Maximum entropy sampling with fixed sample size and unequal probabilities (or Conditional Poisson sampling) is implemented by means of a sequential method.

**Usage**

```
UPmaxentropy(pik)
UPmaxentropypi2(pik)
UPMEqfromw(w,n)
UPMEpikfromq(q)
UPMEpiktildefrompik(pik,eps=1e-6)
UPMEsfromq(q)
UPMEpik2frompikw(pik,w)
```

**Arguments**

n	sample size.
pik	vector of prescribed inclusion probabilities.
eps	tolerance in the Newton's method; by default is 1E-6.
q	matrix of the conditional selection probabilities for the sequential algorithm.
w	parameter vector of the maximum entropy design.

## Details

The maximum entropy sampling maximizes the entropy criterion:

$$I(p) = - \sum_s p(s) \log[p(s)]$$

The main procedure is UPmaxentropy which selects a sample (a vector of 0 and 1) from a given vector of inclusion probabilities. The procedure UPmaxentropypi2 returns the matrix of joint inclusion probabilities from the first-order inclusion probability vector. The other procedures are intermediate steps. They can be useful to run simulations as shown in the examples below. The procedure UPMEpiktildefrompik computes the vector of the inclusion probabilities (denoted pikt) of a Poisson sampling from the vector of the inclusion probabilities of the maximum entropy sampling. The maximum entropy sampling is the conditional design given the fixed sample size. The vector w can be easily obtained by w=pikt/(1-pikt). Once piktilde and w are deduced from pik, a matrix of selection probabilities q can be derived from the sample size n and the vector w via UPMEqfromw. Next, a sample can be selected from q using UPMEsfromq. In order to generate several samples, it is more efficient to compute the matrix q (which needs some calculation), and then to use the procedure UPMEsfromq. The vector of the inclusion probabilities can be recomputed from q using UPMEpikfromq, which also checks the numerical precision of the algorithm. The procedure UPMEpik2frompikw computes the matrix of the joint inclusion probabilities from q and w.

## References

- Chen, S.X., Liu, J.S. (1997). Statistical applications of the Poisson-binomial and conditional Bernoulli distributions, *Statistica Sinica*, 7, 875-892;
- Deville, J.-C. (2000). *Note sur l'algorithme de Chen, Dempster et Liu*. Technical report, CREST-ENSAI, Rennes.
- Matei, A., Tillé, Y. (2005) Evaluation of variance approximations and estimators in maximum entropy sampling with unequal probability and fixed sample size, *Journal of Official Statistics*, Vol. 21, No. 4, p. 543-570.
- Tillé, Y. (2006), *Sampling Algorithms*, Springer.

## Examples

```
#####
## Example 1
#####
# Simple example - sample selection
pik=c(0.07,0.17,0.41,0.61,0.83,0.91)
# First method
UPmaxentropy(pik)
# Second method by using the intermediate procedures
n=sum(pik)
pikt=UPMEpiktildefrompik(pik)
w=pikt/(1-pikt)
q=UPMEqfromw(w,n)
UPMEsfromq(q)
# The matrix of inclusion probabilities
# First method: direct computation from pik
UPmaxentropypi2(pik)
```

```

# Second method: computation from pik and w
UPMEpik2frompikw(pik,w)
#####
## Example 2
#####
# Sample of Belgian municipalities
data(belgianmunicipalities)
attach(belgianmunicipalities)
n=200
pik=inclusionprobabilities(averageincome,n)
s=UPmaxentropy(pik)
#the sample is
as.character(Commune[s==1])
#the joint inclusion probabilities
pi2=UPmaxentropypi2(pik)
rowSums(pi2)/pik/n
#####
## Example 3
#####
# Selection of 200 samples of Belgian municipalities
# Once matrix q is computed, the selection of a sample is very quick.
# Simulations are thus possible.
data(belgianmunicipalities)
attach(belgianmunicipalities)
pik=inclusionprobabilities(averageincome,200)
pik=pik[pik!=1]
n=sum(pik)
pikt=UPMEpiktildefrompik(pik)
w=pikt/(1-pikt)
q=UPMEqfromw(w,n)
N=length(pik)
tt=rep(0,times=N)
#number of simulations; for accurate results, increase the value of 'sim'
sim=200
for(i in 1:sim) tt = tt+UPMEsfromq(q)
tt=tt/sim
sum(abs(tt-pik))

```

---

UPmidzuno

*Midzuno sampling*


---

### Description

Uses the Midzuno's method to select a sample of units (unequal probabilities, without replacement, fixed sample size).

### Usage

```
UPmidzuno(pik,eps=1e-6)
```

**Arguments**

pik                    vector of the inclusion probabilities.  
 eps                    the control value, by default equal to 1e-6.

**Value**

Returns a vector (with elements 0 and 1) of size N, the population size. Each element k of this vector indicates the status of unit k (1, unit k is selected in the sample; 0, otherwise). The value 'eps' is used to control pik (pik>eps & pik < 1-eps).

**References**

Midzuno, H. (1952), On the sampling system with probability proportional to sum of size. *Annals of the Institute of Statistical Mathematics*, 3:99-107.  
 Deville, J.-C. and Tillé, Y. (1998), Unequal probability sampling without replacement through a splitting method, *Biometrika*, 85:89-101.

**See Also**

[UPtille](#)

**Examples**

```
#define the prescribed inclusion probabilities
pik=c(0.2,0.7,0.8,0.5,0.4,0.4)
#select a sample
s=UPmidzuno(pik)
#the sample is
(1:length(pik))[s==1]
```

---

UPmidzunopi2

*Joint inclusion probabilities for Midzuno sampling*


---

**Description**

Computes the joint (second-order) inclusion probabilities for Midzuno sampling.

**Usage**

```
UPmidzunopi2(pik)
```

**Arguments**

pik                    vector of the first-order inclusion probabilities.

**Value**

Returns a  $N \times N$  matrix of the following form: the main diagonal contains the first-order inclusion probabilities for each unit  $k$  in the population; elements  $(k,l)$  are the joint inclusion probabilities of units  $k$  and  $l$ , with  $k$  not equal to  $l$ .  $N$  is the population size.

**References**

Midzuno, H. (1952), On the sampling system with probability proportional to sum of size. *Annals of the Institute of Statistical Mathematics*, 3:99-107.

**See Also**

[UPmidzuno](#)

**Examples**

```
#define the prescribed inclusion probabilities
pik=c(0.2,0.7,0.8,0.5,0.4,0.4)
#matrix of the joint inclusion probabilities
UPmidzunopi2(pik)
```

---

UPminimalsupport

*Minimal support sampling*

---

**Description**

Uses the minimal support method to select a sample of units (unequal probabilities, without replacement, fixed sample size).

**Usage**

```
UPminimalsupport(pik)
```

**Arguments**

`pik`                    vector of the inclusion probabilities.

**Value**

Returns a vector (with elements 0 and 1) of size  $N$ , the population size. Each element  $k$  of this vector indicates the status of unit  $k$  (1, unit  $k$  is selected in the sample; 0, otherwise).

**References**

Deville, J.-C., Tillé, Y. (1998), Unequal probability sampling without replacement through a splitting method, *Biometrika*, 85, 89-101.  
Tillé, Y. (2006), *Sampling Algorithms*, Springer.

**Examples**

```
#####
## Example 1
#####
#defines the prescribed inclusion probabilities
pik=c(0.2,0.7,0.8,0.5,0.4,0.4)
#selects a sample
s=UPminimalsupport(pik)
#the sample is
(1:length(pik))[s==1]
#####
## Example 2
#####
data(belgianmunicipalities)
Tot=belgianmunicipalities$Tot04
name=belgianmunicipalities$Commune
pik=inclusionprobabilities(Tot,200)
#selects a sample
s=UPminimalsupport(pik)
#the sample is
as.vector(name[s==1])
```

UPmultinomial

*Multinomial sampling***Description**

Uses the Hansen-Hurwitz method to select a sample of units (unequal probabilities, with replacement, fixed sample size).

**Usage**

```
UPmultinomial(pik)
```

**Arguments**

pik                    vector of the the inclusion probabilities.

**Value**

Returns a vector (with elements 0 and 1) of size N, the population size. Each element k of this vector indicates the status of unit k (1, unit k is selected in the sample; 0, otherwise).

**References**

Hansen, M. and Hurwitz, W. (1943), On the theory of sampling from finite populations. *Annals of Mathematical Statistics*, 14:333-362.

**See Also**[rmultinom](#)**Examples**

```
#defines the prescribed inclusion probabilities
pik=c(0.2,0.7,0.8,0.5,0.4,0.4)
#selects a sample
s=UPmultinomial(pik)
#the sample is
(1:length(pik))[s==1]
```

UPopips

*Order pips sampling***Description**

Implements order  $\pi ps$  sampling (unequal probabilities, without replacement, fixed sample size).

**Usage**

```
UPopips(lambda,type=c("pareto","uniform","exponential"))
```

**Arguments**

lambda            vector of working inclusion probabilities or target ones.  
type              the type of order sampling (pareto, uniform, exponential).

**Value**

Returns a vector of selected units of size n, the sample size.

**References**

Rosén, B. (1997), Asymptotic theory for order sampling, *Journal of Statistical Planning and Inference*, 62:135-158.  
Rosén, B. (1997), On sampling with probability proportional to size, *Journal of Statistical Planning and Inference*, 62:159-191.

**See Also**[inclusionprobabilities](#)

## Examples

```
#define the working inclusion probabilities
lambda=c(0.2,0.7,0.8,0.5,0.4,0.4)
#draw a Pareto sample
s=UPopips(lambda, type="pareto")
#the sample is
s
```

---

UPpivotal

*Pivotal sampling*

---

## Description

Selects an unequal probability sample using the pivotal method (unequal probabilities, without replacement, fixed sample size).

## Usage

```
UPpivotal(pik, eps=1e-6)
```

## Arguments

pik	vector of the inclusion probabilities.
eps	the control value, by default equal to 1e-6.

## Value

Returns a vector (with elements 0 and 1) of size N, the population size. Each element k of this vector indicates the status of unit k (1, unit k is selected in the sample; 0, otherwise). The value eps is used to control pik ( $pik > eps$  &  $pik < 1-eps$ ).

## References

Deville, J.-C. and Tillé, Y. (1998), Unequal probability sampling without replacement through a splitting method, *Biometrika*, 85:89-101.  
Chauvet, G. and Tillé, Y. (2006). A fast algorithm of balanced sampling. *to appear in Computational Statistics*.  
Tillé, Y. (2006), *Sampling Algorithms*, Springer.

## See Also

[UPrandompivotal](#)

**Examples**

```
#define the prescribed inclusion probabilities
pik=c(0.2,0.7,0.8,0.5,0.4,0.4)
#select a sample
s=UPpivotal(pik)
#the sample is
(1:length(pik))[s==1]
```

UPpoisson

*Poisson sampling***Description**

Draws a Poisson sample using a prescribed vector of first-order inclusion probabilities (unequal probabilities, without replacement, random sample size).

**Usage**

```
UPpoisson(pik)
```

**Arguments**

`pik` vector of the first-order inclusion probabilities.

**Value**

Returns a vector (with elements 0 and 1) of size N, the population size. Each element k of this vector indicates the status of unit k (1, unit k is selected in the sample; 0, otherwise). The value 'eps' is used to control pik ( $pik > eps$  &  $pik < 1 - eps$ ).

**See Also**

[inclusionprobabilities](#)

**Examples**

```
#####
## Example 1
#####
# definition of pik
pik=c(1/3,1/3,1/3)
# selects a sample
s=UPpoisson(pik)
#the sample is
(1:length(pik))[s==1]
#####
## Example 2
#####
data(belgianmunicipalities)
```

```
Tot=belgianmunicipalities$Tot04
name=belgianmunicipalities$Commune
n=200
pik=inclusionprobabilities(Tot,n)
# select a sample
s=UPpoisson(pik)
#the sample is
getdata(name,s)
```

---

UPrandompivotal      *Random pivotal sampling*

---

### Description

Selects a sample using the pivotal method, when the order of the population units is random (unequal probabilities, without replacement, fixed sample size).

### Usage

```
UPrandompivotal(pik,eps=1e-6)
```

### Arguments

pik	vector of the inclusion probabilities.
eps	the control value, by default equal to 1e-6.

### Value

Returns a vector (with elements 0 and 1) of size N, the population size. Each element k of this vector indicates the status of unit k (1, unit k is selected in the sample; 0, otherwise). The value 'eps' is used to control pik (pik>eps and pik<1-eps).

### References

Deville, J.-C. and Tillé, Y. (1998), Unequal probability sampling without replacement through a splitting method, *Biometrika*, 85:89–101.  
Tillé, Y. (2006), *Sampling Algorithms*, Springer.

### See Also

[UPpivotal](#)

### Examples

```
#define the prescribed inclusion probabilities
pik=c(0.2,0.7,0.8,0.5,0.4,0.4)
#select a sample
s=UPrandompivotal(pik)
#the sample is
(1:length(pik))[s==1]
```

---

UPrandomsystematic      *Random systematic sampling*

---

### Description

Selects a sample using the systematic method, when the order of the population units is random (unequal probabilities, without replacement, fixed sample size).

### Usage

```
UPrandomsystematic(pik,eps=1e-6)
```

### Arguments

pik                      vector of the inclusion probabilities.  
eps                      the control value, by default equal to 1e-6.

### Value

Returns a vector (with elements 0 and 1) of size N, the population size. Each element k of this vector indicates the status of unit k (1, unit k is selected in the sample; 0, otherwise). The value 'eps' is used to control pik (pik>eps and pik<1-eps).

### References

Madow, W.G. (1949), On the theory of systematic sampling, II, *Annals of Mathematical Statistics*, 20, 333-354.

### See Also

[UPsystematic](#)

### Examples

```
#define the prescribed inclusion probabilities  
pik=c(0.2,0.7,0.8,0.5,0.4,0.4)  
#select a sample  
s=UPrandomsystematic(pik)  
#the sample is  
(1:length(pik))[s==1]
```

---

UPsampfords

*Sampfords sampling*

---

### Description

Uses the Sampfords method to select a sample of units (unequal probabilities, without replacement, fixed sample size).

### Usage

```
UPsampfords(pik, eps=1e-6)
```

### Arguments

pik	vector of the inclusion probabilities.
eps	the control value, by default equal to 1e-6.

### Value

Returns a vector (with elements 0 and 1) of size N, the population size. Each element k of this vector indicates the status of unit k (1, unit k is selected in the sample; 0, otherwise). The value eps is used to control pik (pik>eps & pik < 1-eps). The sample size must be small with respect to the population size; otherwise, the selection time can be very long.

### References

Sampfords, M. (1967), On sampling without replacement with unequal probabilities of selection, *Biometrika*, 54:499-513.

### See Also

[UPsampfordspi2](#)

### Examples

```
#define the prescribed inclusion probabilities
pik=c(0.2,0.7,0.8,0.5,0.4,0.4)
s=UPsampfords(pik)
#the sample is
(1:length(pik))[s==1]
```

---

UPsampfordpi2      *Joint inclusion probabilities for Sampford sampling*

---

**Description**

Computes the joint (second-order) inclusion probabilities for Sampford sampling.

**Usage**

```
UPsampfordpi2(pik)
```

**Arguments**

pik                      vector of the first-order inclusion probabilities.

**Value**

Returns a  $N \times N$  matrix of the following form: the main diagonal contains the first-order inclusion probabilities for each unit  $k$  in the population; elements  $(k,l)$  are the joint inclusion probabilities of units  $k$  and  $l$ , with  $k$  not equal to  $l$ .  $N$  is the population size.

**References**

Sampford, M. (1967), On sampling without replacement with unequal probabilities of selection, *Biometrika*, 54:499-513.

**See Also**

[UPsampford](#)

**Examples**

```
#define the prescribed inclusion probabilities
pik=c(0.2,0.7,0.8,0.5,0.4,0.4)
#matrix of the joint inclusion probabilities
UPsampfordpi2(pik)
```

---

UPsystematic

*Systematic sampling*


---

**Description**

Uses the systematic method to select a sample of units (unequal probabilities, without replacement, fixed sample size).

**Usage**

```
UPsystematic(pik,eps=1e-6)
```

**Arguments**

pik                    vector of the inclusion probabilities.  
eps                    the control value, by default equal to 1e-6.

**Value**

Returns a vector (with elements 0 and 1) of size N, the population size. Each element k of this vector indicates the status of unit k (1, unit k is selected in the sample; 0, otherwise).

**References**

Madow, W.G. (1949), On the theory of systematic sampling, II, *Annals of Mathematical Statistics*, 20, 333-354.

**See Also**

[inclusionprobabilities](#), [UPrandomsystematic](#)

**Examples**

```
#####
## Example 1
#####
#defines the prescribed inclusion probabilities
pik=c(0.2,0.7,0.8,0.5,0.4,0.4)
#selects a sample
s=UPsystematic(pik)
#the sample is
(1:length(pik))[s==1]
#####
## Example 2
#####
data(belgianmunicipalities)
Tot=belgianmunicipalities$Tot04
name=belgianmunicipalities$Commune
pik=inclusionprobabilities(Tot,200)
```

```
#selects a sample
s=UPsystematic(pik)
#the sample is
as.vector(name[s==1])
# extracts the observed data
getdata(belgianmunicipalities,s)
```

---

UPsystematicpi2

*Joint inclusion probabilities for systematic sampling*

---

### Description

Computes the joint (second-order) inclusion probabilities for systematic sampling.

### Usage

```
UPsystematicpi2(pik)
```

### Arguments

pik                    vector of the first-order inclusion probabilities.

### Value

Returns a  $N \times N$  matrix of the following form: the main diagonal contains the first-order inclusion probabilities for each unit  $k$  in the population; elements  $(k,l)$  are the joint inclusion probabilities of units  $k$  and  $l$ , with  $k$  not equal to  $l$ .  $N$  is the population size.

### References

Madow, W.G. (1949), On the theory of systematic sampling, II, *Annals of Mathematical Statistics*, 20, 333-354.

### See Also

[UPsystematic](#)

### Examples

```
#define the prescribed inclusion probabilities
pik=c(0.2,0.7,0.8,0.5,0.4,0.4)
#matrix of the joint inclusion probabilities
UPsystematicpi2(pik)
```

UPtille

*Tille sampling***Description**

Uses the Tillé's method to select a sample of units (unequal probabilities, without replacement, fixed sample size).

**Usage**

```
UPtille(pik,eps=1e-6)
```

**Arguments**

pik	vector of the inclusion probabilities.
eps	the control value, by default equal to 1e-6.

**Value**

Returns a vector (with elements 0 and 1) of size N, the population size. Each element k of this vector indicates the status of unit k (1, unit k is selected in the sample; 0, otherwise). The value eps is used to control pik ( $pik > eps$  &  $pik < 1 - eps$ ).

**References**

Tillé, Y. (1996), An elimination procedure of unequal probability sampling without replacement, *Biometrika*, 83:238-241.  
 Deville, J.-C. and Tillé, Y. (1998), Unequal probability sampling without replacement through a splitting method, *Biometrika*, 85:89-101.

**See Also**

[UPsystematic](#)

**Examples**

```
#####
## Example 1
#####
#defines the prescribed inclusion probabilities
pik=c(0.2,0.7,0.8,0.5,0.4,0.4)
#selects a sample
s=UPtille(pik)
#the sample is
(1:length(pik))[s==1]
#####
## Example 2
#####
# see the vignette (UPexamples.pdf)
```

---

`UPtillepi2`*Joint inclusion probabilities for Tille sampling*

---

**Description**

Computes the joint (second-order) inclusion probabilities for Tillé sampling.

**Usage**

```
UPtillepi2(pik,eps=1e-6)
```

**Arguments**

<code>pik</code>	vector of the first-order inclusion probabilities.
<code>eps</code>	the control value, by default equal to 1e-6.

**Value**

Returns a  $N \times N$  matrix of the following form: the main diagonal contains the first-order inclusion probabilities for each unit  $k$  in the population; elements  $(k,l)$  are the joint inclusion probabilities of units  $k$  and  $l$ , with  $k$  not equal to  $l$ .  $N$  is the population size. The value `eps` is used to control `pik` (`pik > eps` & `pik < 1-eps`).

**References**

Tillé, Y. (1996), An elimination procedure of unequal probability sampling without replacement, *Biometrika*, 83:238-241.

**See Also**

[UPtille](#)

**Examples**

```
#defines the prescribed inclusion probabilities
pik=c(0.2,0.7,0.8,0.5,0.4,0.4)
pik_joint=UPtillepi2(pik)
#the joint inclusion probabilities
pik_joint
```

---

varest *Variance estimation using the Deville's method*

---

### Description

Computes the variance estimation of an estimator of the population total using the Deville's method.

### Usage

```
varest(Ys,Xs=NULL,pik,w=NULL)
```

### Arguments

Ys	vector of the variable of interest; its length is equal to n, the sample size.
Xs	matrix of the auxiliary variables; for the calibration estimator, this is the matrix of the sample calibration variables.
pik	vector of the first-order inclusion probabilities; its length is equal to n, the sample size.
w	vector of the calibrated weights (for the calibration estimator); its length is equal to n, the sample size.

### Details

The function implements the following estimator:

$$\hat{Var}(Y^s) = \frac{1}{1 - \sum_{k \in s} a_k^2} \sum_{k \in s} (1 - \pi_k) \left( \frac{y_k}{\pi_k} - \frac{\sum_{l \in s} (1 - \pi_l) y_l / \pi_l}{\sum_{l \in s} (1 - \pi_l)} \right)$$

where  $a_k = (1 - \pi_k) / \sum_{l \in s} (1 - \pi_l)$ .

### References

Deville, J.-C. (1993). *Estimation de la variance pour les enquêtes en deux phases*. Manuscript, INSEE, Paris.

### See Also

[calibev](#)

### Examples

```
# Belgian municipalities data base
data(belgianmunicipalities)
attach(belgianmunicipalities)
# Computes the inclusion probabilities
pik=inclusionprobabilities(Tot04,200)
N=length(pik)
n=sum(pik)
```

```

# Defines the variable of interest
y=TaxableIncome
# Draws a Tille sample of size 200
s=UPtille(pik)
# Computes the Horvitz-Thompson estimator
HTestimator(y[s==1],pik[s==1])
# Computes the variance estimation of the Horvitz-Thompson estimator
varest(Ys=y[s==1],pik=pik[s==1])
# for an example using calibration estimator see the 'calibration' vignette
vignette("calibration", package="sampling")

```

---

vartaylor_ratio	<i>Taylor-series linearization variance estimation of a ratio</i>
-----------------	---

---

## Description

Computes the Taylor-series linearization variance estimation of the ratio

$$\frac{\hat{Y}_s}{\hat{X}_s}$$

The estimators in the ratio are Horvitz-Thompson estimators.

## Usage

```
vartaylor_ratio(Ys,Xs,pikls)
```

## Arguments

Ys	vector of the first observed variable; its length is equal to n, the sample size.
Xs	vector of the second observed variable; its length is equal to n, the sample size.
pikls	matrix of the joint inclusion probabilities of the sample units; its dimension is nxn.

## Details

The function implements the following estimator:

$$\widehat{Var}\left(\frac{\hat{Y}_s}{\hat{X}_s}\right) = \sum_{i \in s} \sum_{j \in s} \frac{\pi_{ij} - \pi_i \pi_j}{\pi_{ij}} \frac{\hat{z}_i \hat{z}_j}{\pi_i \pi_j}$$

where  $\hat{z}_i = \frac{Y_{s_i} - \hat{r} X_{s_i}}{\hat{X}_s}$ ,  $\hat{r} = \hat{Y}_s / \hat{X}_s$ ,  $\hat{Y}_s = \sum Y_{s_i} / \pi_i$ ,  $\hat{X}_s = \sum X_{s_i} / \pi_i$ .

## References

Woodruff, R. (1971). *A Simple Method for Approximating the Variance of a Complicated Estimate*, Journal of the American Statistical Association, Vol. 66, No. 334, pp. 411–414.

**Examples**

```

# Belgian municipalities data base
data(belgianmunicipalities)
attach(belgianmunicipalities)
# Computes the inclusion probabilities
pik=inclusionprobabilities(averageincome,100)
# the first variable on the population
Y=Men03
# the second variable on the population
X=Women03
# population size
N=length(pik)
# matrix of joint inclusion prob. on the population for Tille sampling
pikl=UPtillepi2(pik)
# draw a sample using the Tille sampling
s=UPtille(pik)
# inclusion probabilities on the sample
piks=pik[s==1]
# the first observed variable
Ys=Y[s==1]
# the second observed variable
Xs=X[s==1]
# matrix of joint inclusion prob. on the sample
pikls=pikl[s==1,s==1]
# computes the ratio HTestimator(Ys,piks)/HTestimator(Xs,piks) and its estimated variance
vartaylor_ratio(Ys,Xs,pikls)

```

---

writesample

*All possible samples of fixed size*

---

**Description**

Gives a matrix whose rows are the vectors (0 or 1) of all samples of fixed size.

**Usage**

```
writesample(n,N)
```

**Arguments**

n	sample size.
N	population size.

**See Also**

[landingcube](#)

**Examples**

```
# all samples of size 4
# from a population of size 10.
w=writesample(4,10)
# the samples are
t(apply(w,1,function(x) (1:ncol(w))[x==1]))
```

# Index

## \*Topic **datasets**

belgianmunicipalities, 7  
MU284, 29  
swissmunicipalities, 51

## \*Topic **survey**

balancedcluster, 3  
balancedstratification, 4  
balancedtwostage, 5  
calib, 8  
calibev, 10  
checkcalibration, 12  
cleanstrata, 13  
cluster, 14  
disjunctive, 15  
fastflightcube, 16  
getdata, 20  
Hajekestimator, 20  
Hajekstrata, 21  
HTestimator, 22  
HTstrata, 23  
inclusionprobabilities, 24  
inclusionprobastrata, 25  
landingcube, 26  
mstage, 27  
postest, 30  
poststrata, 33  
ratioest, 34  
ratioest\_strata, 35  
regist, 37  
regist\_strata, 38  
rhg, 40  
rhg\_strata, 41  
rmodel, 42  
samplecube, 44  
srswor, 46  
srswor1, 47  
srswr, 48  
strata, 49  
UPbrewer, 52

UPmaxentropy, 53  
UPmidzuno, 55  
UPmidzunopi2, 56  
UPminimalsupport, 57  
UPmultinomial, 58  
UPopips, 59  
UPpivotal, 60  
UPpoisson, 61  
UPrandompivotal, 62  
UPrandomsystematic, 63  
UPsampford, 64  
UPsampfordpi2, 65  
UPsystematic, 66  
UPsystematicpi2, 67  
UPtille, 68  
UPtillepi2, 69  
varest, 70  
vartaylor\_ratio, 71  
writesample, 72

balancedcluster, 3, 6  
balancedstratification, 4, 6, 14, 15, 25  
balancedtwostage, 5  
belgianmunicipalities, 7  
  
calib, 8, 11, 13, 18, 40, 41  
calibev, 9, 10, 70  
checkcalibration, 9, 12, 18  
cleanstrata, 13  
cluster, 14, 20, 28  
  
disjunctive, 15  
  
fastflightcube, 3, 5, 6, 16, 26, 44  
  
gencalib, 9, 17  
getdata, 15, 19, 28, 50  
  
Hajekestimator, 20  
Hajekstrata, 21  
HTestimator, 21, 22, 24

HTstrata, 22, 23

inclusionprobabilities, 14, 24, 27, 49, 59, 61, 66

inclusionprobastrata, 24, 25

landingcube, 3, 5, 6, 26, 44, 72

mstage, 15, 20, 27, 50

MU284, 29

postest, 30, 34

poststrata, 31, 33

ratioest, 34, 36, 38

ratioest\_strata, 35

regist, 35, 37, 39

regist\_strata, 38, 38

rhg, 40, 41, 43

rhg\_strata, 40, 41

rmodel, 42

rmultinom, 49, 59

sample, 47

samplecube, 3, 5, 6, 16, 26, 44

srswor, 20, 46, 48

srswor1, 47

srswr, 47, 48

strata, 15, 20, 28, 49

swissmunicipalities, 51

UPbrewer, 52

UPmaxentropy, 53

UPmaxentropypi2 (UPmaxentropy), 53

UPMEpik2frompikw (UPmaxentropy), 53

UPMEpikfromq (UPmaxentropy), 53

UPMEpiktildefrompik (UPmaxentropy), 53

UPMEqfromw (UPmaxentropy), 53

UPMEsfromq (UPmaxentropy), 53

UPmidzuno, 55, 57

UPmidzunopi2, 56

UPminimalsupport, 57

UPmultinomial, 49, 58

UPpops, 59

UPpivotal, 60, 62

UPpoisson, 61

UPrandompivotal, 60, 62

UPrandomsystematic, 63, 66

UPsampford, 64, 65

UPsampfordpi2, 64, 65

UPsystematic, 20, 53, 63, 66, 67, 68

UPsystematicpi2, 67

UPtille, 23, 56, 68, 69

UPtillepi2, 69

varest, 70

vartaylor\_ratio, 71

writesample, 72