Package ‘samplingR’

June 26, 2023

Type Package

Title Sampling and Estimation Methods

Version 1.0.1

Date 2023-06-25

Maintainer Javier Estévez <javier.estase@gmail.com>

Description Functions to take samples of data, sample size estimation and getting useful estimators such as total, mean, proportion about its population using simple random, stratified, systematic and cluster sampling.

Imports dplyr, methods

License GPL-2

Encoding UTF-8

RoxygenNote 7.2.3

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Javier Estévez [aut, cre, cph]

Repository CRAN

Date/Publication 2023-06-26 11:00:02 UTC

R topics documented:

`cluster.estimator` *Parameter estimation for cluster samples*

Description

Estimates parameters with optional confidence interval for clustered data of similar cluster size.

Usage

```
cluster.estimator(
  N,
  data,
  estimator = c("total", "mean", "proportion", "class total"),
  replace = FALSE,
  alpha
)
```

Arguments

`N`	Number of clusters for the population
`data`	Cluster sample
`estimator`	Estimator to compute. Can be one of "total", "mean", "proportion", "class total". Default is "total".
`replace`	Whether the sample to be taken can have repeated instances or not.
`alpha`	Optional value to calculate estimation error and build 1-alpha

Details

This function admits both grouped and non-grouped by cluster data.
Non-grouped data must have interest variable data in the first column and cluster name each individual belongs to in the last column.
Grouped by cluster data must have interest variable data in the first column, cluster size in the second and the cluster name in the last column. Interest values of grouped data must reflect the total value of each cluster.

Value

A list containing different interest values:

- estimator
- variance
- sampling.error
- estimation.error
- confint

Examples

```
d<-cbind(rnorm(500, 50, 20), rep(c(1:50),10)) #Non-grouped data
sample<-cluster.sample(d, n=10) #Non-grouped sample
sampleg<-aggregate(sample[,1], by=list(Category=sample[,2]), FUN=sum)
sampleg<-cbind(sampleg[,2], rep(10,10), sampleg[,1]) #Same sample but with grouped data
sum(d[,1])
cluster.estimator(N=50, data=sample, estimator="total", alpha=0.05)
cluster.estimator(N=50, data=sampleg, estimator="total", alpha=0.05)
```

`cluster.sample` *Cluster sample for non-grouped data.*

Description

Retrieves a sample of clusters for data which interest variable values are not grouped by cluster.

Usage

```
cluster.sample(data, n, replace = FALSE)
```

Arguments

`data`	Matrix or data.frame containing the population data in first column and the cluster it belongs to in the last column.
`n`	Number of clusters of the returning sample.
`replace`	Whether the sample to be taken can have repeated clusters or not.

Details

#' If your data is grouped by cluster use `srs.sample` function to retrieve your sample. Remember grouped by cluster data must have interest variable data in the first column, cluster size in the second and the cluster name in the last column. Interest values of grouped data must reflect the total value of each cluster.

Value

Data frame of a clustered sample

Examples

```
data<-cbind(rnorm(500, 50, 20), rep(c(1:50),10))
sample<-cluster.sample(data, 10);sample
```

`cluster.samplesize` *Sample size estimation on cluster sampling*

Description

Calculates the required sample size in order to achieve an absolute sampling error less or equal to the specified for an specific estimator and an optional confidence interval in cluster sampling.

Usage

```
cluster.samplesize(
  N,
  data,
  error,
  alpha,
  estimator = c("total", "mean", "proportion", "class total"),
  replace = FALSE
)
```

Arguments

`N`	Number of clusters in the population.
`data`	Dataset.
`error`	Sampling error.
`alpha`	Significance level to obtain confidence intervals.
`estimator`	The estimator to be estimated. Default is "total".
`replace`	Whether the samples to be taken can have repeated instances or not.

Details

This function admits both grouped and non-grouped by cluster data.
Non-grouped data must have interest variable data in the first column and cluster name each individual belongs to in the last column.
Grouped by cluster data must have interest variable data in the first column, cluster size in the second and the cluster name in the last column. Interest values of grouped data must reflect the total value of each cluster.

Value

Number of clusters to be taken.

Examples

```
d<-cbind(rnorm(500, 50, 20), rep(c(1:50),10)) #Non-grouped data
sample<-cluster.sample(d, n=10) #Non-grouped sample
sampleg<-aggregate(sample[,1], by=list(Category=sample[,2]), FUN=sum)
sampleg<-cbind(sampleg[,2], rep(10,10), sampleg[,1]) #Sample sample with grouped data

#Cluster size to be taken for estimation
cluster.samplesize(N=50, data=sample, error=500, estimator="total", replace=TRUE)

newsample<-cluster.sample(d, n=26) #New sample for estimation
sum(d[,1])
cluster.estimator(N=50, data=newsample, estimator="total", alpha=0.05, replace=TRUE)
cluster.estimator(N=50, data=sampleg, estimator="total", alpha=0.05)
```

`srs.domainestimator` *Simple Random Sample parameter estimation of domains.*

Description

Function to make estimations of diferent parameters on a given domain based on a Simple Random Sample.

Usage

```
srs.domainestimator(
  Nh,
  data,
  estimator = c("total", "mean", "proportion", "class total"),
  domain,
  replace = FALSE,
  alpha
)
```

Arguments

`Nh`	Number of instances of the data set domain.
`data`	Sample of the data. It must constain a column with the data to estimate and a second column with the domain of each instance.
`estimator`	One of "total", "mean". Default is "total".
`domain`	Domain of the sample from which parameter estimation will be done.
`replace`	Whether the sample to be taken can have repeated instances or not.
`alpha`	Optional value to calculate estimation error and build 1-alpha confidence interval.

Details

Data columns must be arranged with interest values on the first column and domain values on the last column.
Domain parameter can be either numeric or character and must be equal to one of the values of the domain column of data.

Value

A list containing different interest values:

- estimator
- variance
- sampling.error
- estimation.error
- confint

References

Pérez, C. (1999) Técnicas de muestreo estadístico. Teoría, práctica y aplicaciones informáticas. 193-195

Examples

```
data<-cbind(rnorm(500, 50, 20), rep(c(1:2),250))
sample<-data[srs.sample(500, 100),]
sum(data[which(data[,-1]==1),1])
srs.domainestimator(Nh = 250, data = sample, estimator="total", domain=1)
```

---

srs.estimator *Simple Ramdom Sampling parameter estimation.*

Description

Function to make estimations of diferent parameters based on a Simple Random Sample.

Usage

```
srs.estimator(
  N,
  data,
  estimator = c("total", "mean", "proportion", "class total"),
  replace = FALSE,
  alpha
)
```

Arguments

`N`	Number of instances of the data set.
`data`	Sample of the data. It must only contain a single column of the data to estimate.
`estimator`	Estimator to compute. Can be one of "total", "mean", "proportion", "class total". Default is "total".
`replace`	Whether the sample has been taken with replacement or not.
`alpha`	Optional value to calculate estimation error and build 1-alpha confidence interval.

Value

A list containing different interest values:

- estimator
- variance
- sampling.error
- estimation.error
- confint

Examples

```
data<-rnorm(200, 100, 20)
sample<-data[srs.sample(200, 50)]
tau<-sum(data);tau
srs.estimator(200, sample, "total", alpha=0.05)


mu<-mean(data);mu
srs.estimator(200, sample, "mean", alpha=0.05)
```

`srs.sample` *Simple Random Sample*

Description

With this function you receive a simple random sample consisting on a list of the instances index

Usage

```
srs.sample(N, n, replace = FALSE, data)
```

Arguments

`N`	Number of instances of the data set.
`n`	Number of instances of the returning sample.
`replace`	Whether the sample to be taken can have repeated instances or not.
`data`	Optional matrix or data.frame containing the population data. If specified an object of same class as data will be returned with sample instances.

Value

List of size n with numbers from 1 to N indicating the index of the data set's instances to be taken.

Examples

```
srs.sample(10,3)


data<-matrix(data=c(1:24), nrow=8)
N<-dim(data)[1]
sample<-srs.sample(N, 3, data = data)
sample
```

`srs.samplesize` *Simple Random Sample size.*

Description

Calculates the required sample size in order to achieve a relative or absolute sampling error less or equal to the specified for an specific estimator and an optional confidence interval in simple random sampling.

Usage

```
srs.samplesize(
  N,
  var,
  error,
  alpha,
  estimator = c("total", "mean", "proportion", "class total"),
  p,
  mean,
  replace = FALSE,
  relative = FALSE
)
```

Arguments

N	Number of instances of the data set.
var	Estimated quasivariance.
error	Sampling error
alpha	Significance level to obtain confidence intervals.
estimator	One of "total", "proportion", "mean", "class total". Default is "total"
p	Estimated proportion. If estimator is not "proportion" or "class total" it will be ignored.
mean	Estimated mean. If relative=FALSE it will be ignored.
replace	Whether the sample to be taken can have repeated instances or not.
relative	Whether the specified error is relative or not.

Details

If the sample size result is not a whole number the number returned is the next whole number so srs.samplesize>=n is satisfied.
To estimate sample size of estimators "total" and "mean" estimated quasivariance must be provided. If the error is relative then estimated mean must also be provided.
To estimate sample size of estimator "proportion" and "class total" estimated proportion must be provided. If p is not specified sample size will be estimated based on worst-case scenario of p=0.5.
N must be always be provided for calculations.

Value

Number of instances of the sample to be taken.

Examples

```
data<-rnorm(200, 100, 20)
n<-srs.samplesize(200, var(data), estimator="total", error=400, alpha=0.05);n
sample<-data[srs.sample(200, n)]
srs.estimator(200, sample, "total", alpha=0.05)$sampling.error
```

---

strata.allocation *Strata allocation given a sample size*

Description

Function to allocate the number of samples to be taken for each strata given the total sample size and the strata.allocation method. The number of allocations returned will be equal to the length of the parameters.

Usage

```
strata.allocation(
  Nh,
  n,
  var,
  alloc = c("unif", "prop", "min", "optim"),
  C,
  cini,
  ch
)
```

Arguments

`Nh`	Vector of population strata sizes.
`n`	Sample size
`var`	Vector of strata variances.
`alloc`	The allocation method to be used. Default is "unif".
`C`	Total study cost.
`cini`	Overhead study cost.
`ch`	Vector of costs to take an individual from a strata for the sample.

Details

alloc="optim" is the only that requires cost function data. Total study and overhead study costs are optional. If given allocation will be done so total study cost is not surpassed.s

Value

Vector of strata sample sizes.

Examples

```
strata.allocation(Nh=rep(125,4), n=100, alloc="unif") #25, 25, 25, 25
strata.allocation(Nh=c(100, 50, 25), n=100, alloc="prop")
```

`strata.estimator` *Parameter estimation of stratified data*

Description

Function to make estimations of diferent parameters based on a stratified sample.

Usage

```
strata.estimator(
  N,
  Nh,
  data,
  estimator = c("total", "mean", "proportion", "class total"),
  replace = FALSE,
  alpha
)
```

Arguments

`N`	Population size.
`Nh`	Size of each population strata.
`data`	Stratified sample.
`estimator`	Estimator to compute. Can be one of "total", "mean", "proportion", "class total". Default is "total".
`replace`	Whether the sample to be taken can have repeated instances or not.
`alpha`	Optional value to calculate estimation error and build 1-alpha

Details

Nh length must be equal to number of strata in data.
data is meant to be a returned object of `strata.sample` function.

Value

A list containing different interest values:

- estimator
- variance
- sampling.error
- estimation.error
- confint

`strata.sample` *Stratified sample*

Description

With this function you receive a sample of each strata within your data with specified size for each strata.

Usage

```
strata.sample(data, n, replace = FALSE)
```

Arguments

`data`	Population data consisting of a number of columns of data and a last column specifying the strata each instance belongs to.
`n`	Numeric array of sample sizes for each strata to be taken.
`replace`	Whether the sample to be taken can have repeated instances or not.

Details

n length must be equal to number of strata in data.
On return list each strata sample can be accessed calling object$strataname where strataname are values of the last column of the original data.

Value

A list containing one strata sample per index.

Examples

```
data<-cbind(rnorm(500, 50, 20), rep(c("clase 1", "clase 2","clase 3","clase4"),125))
strata.sample(data=data, n=c(10,20,30,40))
```

`strata.samplesize` *Sample size estimation on stratified sampling*

Description

Calculates the required sample size in order to achieve an absolute or relative sampling error less or equal to the specified for an specific estimator and an optional confidence interval in stratified sampling.

Usage

```
strata.samplesize(
  Nh,
  var,
  error,
  alpha,
  estimator = c("total", "mean", "proportion", "class total"),
  alloc = c("prop", "min", "optim"),
  ch,
  p,
  mean,
  replace = FALSE,
  relative = FALSE
)
```

Arguments

`Nh`	Vector of population strata sizes.
`var`	Vector of estimated strata variances.
`error`	Sampling error.
`alpha`	Significance level to obtain confidence intervals.
`estimator`	The estimator to be estimated. Default is "total".
`alloc`	The allocation to be used when taking samples. Default is "prop".
`ch`	Vector of cost per strata to select an individual for the sample.
`p`	Estimated population proportion. If estimator is not "proportion" or "class total" it will be ignored.
`mean`	Estimated population mean. If relative=FALSE it will be ignored.
`replace`	Whether the samples to be taken can have repeated instances or not.
`relative`	Whether the specified error is relative or not.

Details

With "proportion" and "class total" estimators variance vector must contain `var` return values equal to $\frac{Nh}{(Nh-1)} p * (1-p)$ values.

Value

Number of instances of the sample to be taken.

Examples

```
strata.samplesize(c(120,100,110,50), c(458, 313,407,364), error=5, alpha=0.05, "mean", "prop")
```

`strata.samplesize.cost` *Strata sample size by costs function*

Description

This function returns the total sample size given a costs function consisting on the fixed total study cost, overhead study cost and a vector of costs by strata. can be given so the allocation is calculated to not exceed the total study cost.

Usage

```
strata.samplesize.cost(
  Nh,
  var,
  C,
  cini,
  ch,
  alloc = c("unif", "prop", "optim")
)
```

Arguments

`Nh`	Vector of population strata sizes.
`var`	Vector of strata variance values.
`C`	Total study cost.
`cini`	Overhead study cost.
`ch`	Vector of costs to take an individual from a strata for the sample.
`alloc`	The allocation method to be used. Default is "unif".

Details

Strata variance values are only necessary for optim allocation.

Value

Sample size.

Examples

```
strata.samplesize.cost(Nh=c(100,500,200), C=1000, cini=70, ch=c(9,5,12), alloc="prop")
```

`syst.all.samples` *Systematic samples*

Description

Returns all possible systematic samples of size n

Usage

```
syst.all.samples(data, n)
```

Arguments

`data`	Population data
`n`	Sample size

Value

List with a sample per entrance

Examples

```
data<-c(1,3,5,2,4,6,2,7,3)
syst.all.samples(data, 3)
```

`syst.anova` *Analysis of variance of population data*

Description

Analysis of variance of population data

Usage

```
syst.anova(data, n)
```

Arguments

`data`	Population data
`n`	Sample size

Value

Summary

Examples

```
data<-c(1,3,5,2,4,6,2,7,3)
syst.anova(data,3)
```

`syst.estimator` *Parameter estimation on a systematic sample*

Description

Parameter estimation on a systematic sample

Usage

```
syst.estimator(
  N,
  sample,
  estimator = c("total", "mean", "proportion", "class total"),
  method = c("srs", "strata", "syst"),
  alpha,
  data,
  t
)
```

Arguments

`N`	Population size
`sample`	Vector containing the systematic sample
`estimator`	Estimator to compute. Can be one of "total", "mean", "proportion", "class total". Default is "total".
`method`	Method of variance estimation. Can be one of "srs", "strata", "syst".
`alpha`	Optional value to calculate estimation error and build 1-alpha confidence interval.
`data`	Population data.
`t`	Number of systematic samples to take with interpenetrating samples method.

Details

Variance estimation has no direct formula in systematic sampling, thus estimation method must be done. Refer to `syst.intracorr` and `syst.intercorr` functions details for more information.
"syst" method uses interpenetrating samples method in which t systematic samples of size=$\frac{n}{t}$ are taken to estimate. $\frac{n}{t}$ must be even.
By choosing the start at random for all the samples they can be considered random taken. With this method population data and t must be given.

Value

A list containing different interest values:

- estimator
- variance
- sampling.error
- estimation.error
- confint

Examples

```
data<-c(1,3,5,2,4,6,2,7,3)
sample<-syst.sample(9, 3, data)
syst.estimator(N=9, sample, "mean", "srs", 0.05)
```

`syst.intercorr` *Stratified correlation coefficient*

Description

Stratified correlation coefficient

Usage

```
syst.intercorr(N, n, data)
```

Arguments

`N`	Population size
`n`	Sample size
`data`	Population data

Details

This value serves as a comparison between systematic and stratified sampling precision.
At value=1 the systematic precision is minimum. At value=0 both sampling methods precision are equal. At value= $\frac{-1}{n-1}$ systematic precision is maximum.
Summarising at values between 1 and 0 stratified sampling estimation has more precision than systematic, so method="strata" should be set at `syst.estimator`. The other way method="syst" of interpenetrating samples method is better.

Value

Correlation coefficient

Examples

```
data<-c(1,3,5,2,4,6,2,7,3)
syst.intercorr(9,3,data)  #0.09022556
```

`syst.intracorr` *Intraclass correlation coefficient*

Description

Intraclass correlation coefficient

Usage

```
syst.intracorr(N, n, data)
```

Arguments

`N`	Population size
`n`	Sample size
`data`	Population data

Details

This value serves as a comparison between systematic and simple random sampling precision.
At value=1 the systematic precision is minimum. At value=0 both sampling methods precision are equal. At value= $\frac{-1}{n-1}$ systematic precision is maximum.
Summarising at values between 1 and 0 simple random sampling estimation has more precision than systematic, so method="srs" should be set at `syst.estimator`. The other way method="syst" of interpenetrating samples method is better.

Value

Intraclass correlation

Examples

```
data<-c(1,3,5,2,4,6,2,7,3)
syst.intracorr(9, 3, data)  #0.34375 example 1
```

`syst.sample` *Systematic sampling sample*

Description

Retrieves a $\frac{N}{n}$ systematic sample

Usage

```
syst.sample(N, n, data)
```

Arguments

`N`	Population size.
`n`	Sample size
`data`	Optional data of the population.

Details

If $\frac{N}{n}$ is not an even number a 1 in floor($\frac{N}{n}$) sample will be taken.

Value

Vector of size n with numbers from 1 to N indicating the index samples to be taken. If data is provided then the instances will be returned.

Examples

```
data<-runif(40)
syst.sample(40,8, data)
```

Index