Package 'NonProbEst'

July 21, 2025

Type Package

Title Estimation in Nonprobability Sampling

Version 0.2.4

Author Luis Castro Martín <luiscastro193@gmail.com>, Ramón Ferri García <rferri@ugr.es> and María del Mar Rueda <mrueda@ugr.es>

Maintainer Luis Castro Martín <luiscastro193@gmail.com>

Description Different inference procedures are proposed in the literature to correct for selection bias that might be introduced with non-random selection mechanisms. A class of methods to correct for selection bias is to apply a statistical model to predict the units not in the sample (super-population modeling). Other studies use calibration or Statistical Matching (statistically match nonprobability and probability samples). To date, the more relevant methods are weighting by Propensity Score Adjustment (PSA).
The Propensity Score Adjustment method was originally developed to construct weights by estimating response probabilities and using them in Horvitz–Thompson type estimators. This method is usually used by combining a non-probability sample with a reference sample to construct propensity models for the non-probability sample. Calibration can be used in a posterior way to adding information of auxiliary variables.
Propensity scores in PSA are usually estimated using logistic regression models. Machine learning classification algorithms can be used as alternatives for logistic regression as a technique to estimate propensities.
The package 'NonProbEst' implements some of these methods and thus provides a wide options to work with data coming from a non-probabilistic sample.

License GPL (>= 2)

Encoding UTF-8

LazyData true

Imports caret, sampling, e1071, glmnet, Matrix

RoxygenNote 7.1.0

NeedsCompilation no

Repository CRAN

Date/Publication 2020-06-03 12:10:03 UTC

Contents

`calib_weights` *Weights of the calibration estimator*

Description

Calculates the calibration weights from a disjunct matrix of covariates, a vector of population totals and a vector of initial weights.

Usage

```
calib_weights(Xs, totals, initial_weights, N, ...)
```

Arguments

`Xs`	Matrix of calibration variables.
`totals`	A vector containing population totals for each column (class) of the calibration variables matrix.
`initial_weights`	A vector containing the initial weights for each individual.
`N`	Integer indicating the population size.
`...`	Further arguments to be passed to the ‘calib‘ function from the ‘sampling‘ package.

Details

The function uses the ‘calib‘ function from the ‘sampling‘ package for the estimation of g-weights, which are multiplied by the initial weights to obtain the final calibration weights. The initial weights can be calculated previously from the propensities for any of the implemented methods (see functions `lee_weights`, `sc_weights`, `valliant_weights`, `vd_weights`). The population size is used to scale said initial weights so they are easier to calibrate.

Value

A vector with the corresponding weights.

Examples

```
n = nrow(sampleNP)
N = 50000
language_total = 45429
covariates = c("education_primaria", "education_secundaria",
   "age", "sex")
pi = propensities(sampleNP, sampleP, covariates, algorithm = "glm", smooth = FALSE)
wi = sc_weights(pi$convenience)
calib_weights(sampleNP$language, language_total, wi, N, method = "raking")
```

`confidence_interval` *Confidence interval*

Description

Calculates the confidence interval for the estimator considered.

Usage

```
confidence_interval(estimation, std_dev, confidence = 0.95)
```

Arguments

`estimation`	A numeric value specifying the point estimation.
`std_dev`	A numeric value specifying the standard deviation of the point estimation.
`confidence`	A numeric value between 0 and 1 specifying the confidence level, taken as 1 - alpha (1 - Type I error). By default, its value is 0.95.

Value

A vector containing the lower and upper bounds.

Examples

```
covariates = c("education_primaria","education_secundaria",
"age", "sex")
pi = propensities(sampleNP, sampleP, covariates, algorithm = "glm", smooth = FALSE)
psa_weights = sc_weights(pi$convenience)
N = 50000
Y_est = total_estimation(sampleNP, psa_weights, estimated_vars = "vote_pens", N = N)
VY_est = fast_jackknife_variance(sampleNP, psa_weights,
   estimated_vars = "vote_pens") * N^2
confidence_interval(Y_est, sqrt(VY_est), confidence = 0.90)
```

`fast_jackknife_variance` *Calculates Jackknife variance without reweighting*

Description

Calculates the variance of a given estimator by Leave-One-Out Jackknife (Quenouille, 1956) with the original adjusted weights.

Usage

```
fast_jackknife_variance(sample, weights, estimated_vars, N = NULL)
```

Arguments

`sample`	A data frame containing the sample.
`weights`	A vector containing the pre-calculated weights.
`estimated_vars`	A string vector specifying the variables for which the estimators' variance are to be estimated.
`N`	Integer indicating the population size. Optional.

Details

The variance estimation is performed by eliminating an individual at each iteration with its corresponding weight and estimating the mean of the corresponding subsample, which is further used in the Jackknife formula as the usual procedure. The calculation of variance estimates through this procedure might take less computation time but also might not take into account the variance of the weighting method.

Value

A vector containing the resulting variance for each variable.

References

Quenouille, M. H. (1956). *Notes on bias in estimation.* Biometrika, 43(3/4), 353-360.

Examples

```
covariates = c("education_primaria", "education_secundaria")
data_propensities = propensities(sampleNP, sampleP, covariates)
psa_weights = sc_weights(data_propensities$convenience)
fast_jackknife_variance(sampleNP, psa_weights, c("vote_pens"), 50000)
```

---

generic_jackknife_variance

Calculates Jackknife variance with reweighting for an arbitrary estimator

---

Description

Calculates the variance of a given estimator by Leave-One-Out Jackknife (Quenouille, 1956) with reweighting in each iteration.

Usage

```
generic_jackknife_variance(sample, estimator, N = NULL)
```

Arguments

`sample`	Data frame containing the non-probabilistic sample.
`estimator`	Function that, given a sample as a parameter, returns an estimation.
`N`	Integer indicating the population size. Optional.

Details

The estimation of the variance requires a recalculation of the estimates in each iteration which might involve weighting adjustments, leading to an increase in computation time. It is expected that the estimated variance captures the weighting adjustments' variability and the estimator's variability.

Value

The resulting variance.

References

Quenouille, M. H. (1956). *Notes on bias in estimation.* Biometrika, 43(3/4), 353-360.

Examples

```
covariates = c("education_primaria", "education_secundaria",
   "age", "sex", "language")
if (is.numeric(sampleNP$vote_gen))
   sampleNP$vote_gen = factor(sampleNP$vote_gen, c(0, 1), c('F', 'T'))
vote_gen_estimator = function(sample) {
   model_based(sample, population, covariates,
      "vote_gen", positive_label = 'T', algorithm = 'glmnet')
}
generic_jackknife_variance(sampleNP, vote_gen_estimator)
```

`jackknife_variance` *Calculates Jackknife variance with reweighting for PSA*

Description

Calculates the variance of PSA by Leave-One-Out Jackknife (Quenouille, 1956) with reweighting in each iteration.

Usage

```
jackknife_variance(
  estimated_vars,
  convenience_sample,
  reference_sample,
  covariates,
  N = NULL,
  algorithm = "glm",
  smooth = FALSE,
  proc = NULL,
  trControl = trainControl(classProbs = TRUE),
  weighting.func = "sc",
  g = 5,
  calib = FALSE,
  calib_vars = NULL,
  totals = NULL,
  args.calib = NULL,
  ...
)
```

Arguments

`estimated_vars`	A string vector specifying the variables for which the estimators' variance are to be estimated.
`convenience_sample`	Data frame containing the non-probabilistic sample.

`reference_sample`	Data frame containing the probabilistic sample.
`covariates`	String vector specifying the common variables to use for training.
`N`	Integer indicating the population size. Optional.
`algorithm`	A string specifying which classification or regression model to use (same as caret's method). By default, its value is "glm" (logistic regression).
`smooth`	A logical value; if TRUE, propensity estimates pi_i are smoothed applying the formula (1000*pi_i + 0.5)/1001
`proc`	A string or vector of strings specifying if any of the data preprocessing techniques available in train function from 'caret' package should be applied to data prior to the propensity estimation. By default, its value is NULL and no preprocessing is applied.
`trControl`	A trainControl specifying the computational nuances of the train function.
`weighting.func`	A string specifying which function should be used to compute weights from propensity scores. Available functions are the following: • `sc` calls sc_weights. • `valliant` calls valliant_weights. • `lee` calls lee_weights. • `vd` calls vd_weights.
`g`	If `weighting.func = "lee"` or `weighting.func = "vd"`, this element specifies the number of strata to use; by default, its value is 5.
`calib`	A logical value; if TRUE, PSA weights are used as initial weights for calibration. By default, its value is FALSE.
`calib_vars`	A string or vector of strings specifying the variables to be used for calibration. By default, its value is NULL.
`totals`	A vector containing population totals for each column (class) of the calibration variables matrix. Ignored if `calib` is set to FALSE.
`args.calib`	A list containing further arguments to be passed to the calib_weights function.
`...`	Further parameters to be passed to the train function.

Details

The estimation of the variance requires a recalculation of the estimates in each iteration which might involve weighting adjustments, leading to an increase in computation time. It is expected that the estimated variance captures the weighting adjustments' variability and the estimator's variability.

Value

The resulting variance.

References

Quenouille, M. H. (1956). *Notes on bias in estimation.* Biometrika, 43(3/4), 353-360.

Examples

```
#A simple example without calibration and default parameters
covariates = c("education_primaria", "education_secundaria")
jackknife_variance("vote_pens",sampleNP, sampleP, covariates)

#An example with linear calibration and default parameters
covariates = c("education_primaria", "education_secundaria")
calib_vars = c("age", "sex")
totals = c(2544377, 24284)

jackknife_variance("vote_pens",sampleNP, sampleP, covariates,
calib = T, calib_vars, totals, args.calib = list(method = "linear"))
```

`lee_weights` *Calculates Lee weights*

Description

Computes weights from propensity estimates using the propensity stratification design weights averaging formula introduced in Lee (2006) and Lee and Valliant (2009).

Usage

```
lee_weights(convenience_propensities, reference_propensities, g = 5)
```

Arguments

`convenience_propensities`	A vector with the propensities associated with the convenience sample.
`reference_propensities`	A vector with the propensities associated with the reference sample.
`g`	The number of strata to use; by default, its value is 5.

Details

The function takes the vector of propensities $\pi(x)$ and calculates the weights to be applied in the Horvitz-Thompson estimator using the formula that can be found in Lee (2006) and Lee and Valliant (2009). The vector of propensities is divided in *g* strata (ideally five according to Cochran, 1968) aiming to have individuals with similar propensities in each strata. After the stratification, weight is calculated as follows for an individual *i*:

$$w_i = \frac{n_r(g_i)/n_r}{n_v(g_i)/n_v}$$

where g_i represents the strata to which *i* belongs, $n_r(g_i)$ and $n_v(g_i)$ are the number of individuals in the g_i strata from the reference and the convenience sample respectively, and n_r and n_v are the sample sizes for the reference and the convenience sample respectively.

Value

A vector with the corresponding weights.

References

Lee, S. (2006). *Propensity score adjustment as a weighting scheme for volunteer panel web surveys.* Journal of official statistics, 22(2), 329.

Lee, S., & Valliant, R. (2009). *Estimation for volunteer panel web surveys using propensity score adjustment and calibration adjustment.* Sociological Methods & Research, 37(3), 319-343.

Cochran, W. G. (1968). *The Effectiveness of Adjustment by Subclassification in Removing Bias in Observational Studies.* Biometrics, 24(2), 295-313

Examples

```
covariates = c("education_primaria", "education_secundaria")
data_propensities = propensities(sampleNP, sampleP, covariates)
lee_weights(data_propensities$convenience, data_propensities$reference)
```

`matching` *Predicts unknown responses by matching*

Description

It uses the matching method introduced by Rivers (2007). The idea is to model the relationship between y_k and x_k using the convenience sample in order to predict y_k for the reference sample. You can then predict the total using the ‘total_estimation‘ method.

Usage

```
matching(
  convenience_sample,
  reference_sample,
  covariates,
  estimated_var,
  positive_label = NULL,
  algorithm = "glm",
  proc = NULL,
  ...
)
```

Arguments

`convenience_sample`	Data frame containing the non-probabilistic sample.
`reference_sample`	Data frame containing the probabilistic sample.

`covariates`	String vector specifying the common variables to use for training.
`estimated_var`	String specifying the variable to estimate.
`positive_label`	String specifying the label to be considered positive if the estimated variable is categorical. Leave it as the default NULL otherwise.
`algorithm`	A string specifying which classification or regression model to use (same as caret's method).
`proc`	A string or vector of strings specifying if any of the data preprocessing techniques available in train function from 'caret' package should be applied to data prior to the propensity estimation. By default, its value is NULL and no preprocessing is applied.
`...`	Further parameters to be passed to the train function.

Details

Training of the models is done via the 'caret' package. The algorithm specified in `algorithm` must match one of the names in the list of algorithms supported by 'caret'. If the estimated variable is categorical, probabilities are returned.

Value

A vector containing the estimated responses for the reference sample.

References

Rivers, D. (2007). *Sampling for Web Surveys*. Presented in Joint Statistical Meetings, Salt Lake City, UT.

Examples

```
#Simple example with default parameters
N = 50000
covariates = c("education_primaria", "education_secundaria")
if (is.numeric(sampleNP$vote_gen))
   sampleNP$vote_gen = factor(sampleNP$vote_gen, c(0, 1), c('F', 'T'))
estimated_votes = data.frame(
   vote_gen = matching(sampleNP, sampleP, covariates, "vote_gen", 'T')
)
total_estimation(estimated_votes, N / nrow(estimated_votes), c("vote_gen"), N)
```

`mean_estimation` *Estimates the population means*

Description

Estimates the means for the specified variables measured in a sample given some pre-calculated weights.

Usage

```
mean_estimation(sample, weights, estimated_vars, N = NULL)
```

Arguments

`sample`	A data frame containing the sample with the variables for which the means are to be calculated.
`weights`	A vector of pre-calculated weights.
`estimated_vars`	String vector specifying the variables in the sample to be estimated.
`N`	An integer specifying the population size (optional).

Value

A vector with the corresponding estimations.

Examples

```
covariates = c("education_primaria", "education_secundaria")
data_propensities = propensities(sampleNP, sampleP, covariates)
psa_weights = sc_weights(data_propensities$convenience)
mean_estimation(sampleNP, psa_weights, c("vote_pens"))
```

`model_assisted` *Calculates a model assisted estimation*

Description

It uses the model assisted estimator introduced by Särndal et al. (1992).

Usage

```
model_assisted(
  sample_data,
  weights,
  full_data,
  covariates,
  estimated_var,
  estimate_mean = FALSE,
  positive_label = NULL,
  algorithm = "glm",
  proc = NULL,
  ...
)
```

Arguments

`sample_data`	Data frame containing the sample.
`weights`	Vector containing the sample weights.
`full_data`	Data frame containing all the individuals contained in the population.
`covariates`	String vector specifying the common variables to use for training.
`estimated_var`	String specifying the variable to estimate.
`estimate_mean`	Boolean specifying whether the mean estimation should be returned. Otherwise, the total estimation is returned by default.
`positive_label`	String specifying the label to be considered positive if the estimated variable is categorical. Leave it as the default NULL otherwise.
`algorithm`	A string specifying which classification or regression model to use (same as caret's method).
`proc`	A string or vector of strings specifying if any of the data preprocessing techniques available in train function from ‘caret‘ package should be applied to data prior to the propensity estimation. By default, its value is NULL and no preprocessing is applied.
`...`	Further parameters to be passed to the train function.

Details

Training of the models is done via the ‘caret‘ package. The algorithm specified in `algorithm` must match one of the names in the list of algorithms supported by ‘caret‘.

Value

The population total estimation (or mean if specified by the ‘estimate_mean‘ parameter).

References

Särndal, C. E., Swensson, B., & Wretman, J. (1992). *Model assisted survey sampling.* Springer, New York.

Examples

```
#Simple example
covariates = c("education_primaria", "education_secundaria",
   "age", "sex", "language")
if (is.numeric(sampleNP$vote_gen))
   sampleNP$vote_gen = factor(sampleNP$vote_gen, c(0, 1), c('F', 'T'))
model_assisted(sampleNP, nrow(population) / nrow(sampleNP),
   population, covariates, "vote_gen", positive_label = 'T', algorithm = 'glmnet')
```

`model_based` *Calculates a model based estimation*

Description

It uses the model based estimator. The idea in order to estimate the population total is to add the sample responses and the predicted responses for the individuals not contained in the sample. See for example Valliant et al. (2000).

Usage

```
model_based(
  sample_data,
  full_data,
  covariates,
  estimated_var,
  estimate_mean = FALSE,
  positive_label = NULL,
  algorithm = "glm",
  proc = NULL,
  ...
)
```

Arguments

`sample_data`	Data frame containing the sample.
`full_data`	Data frame containing all the individuals contained in the population.
`covariates`	String vector specifying the common variables to use for training.
`estimated_var`	String specifying the variable to estimate.
`estimate_mean`	Boolean specifying whether the mean estimation should be returned. Otherwise, the total estimation is returned by default.
`positive_label`	String specifying the label to be considered positive if the estimated variable is categorical. Leave it as the default NULL otherwise.
`algorithm`	A string specifying which classification or regression model to use (same as caret's method).
`proc`	A string or vector of strings specifying if any of the data preprocessing techniques available in train function from 'caret' package should be applied to data prior to the propensity estimation. By default, its value is NULL and no preprocessing is applied.
`...`	Further parameters to be passed to the train function.

Details

Training of the models is done via the 'caret' package. The algorithm specified in `algorithm` must match one of the names in the list of algorithms supported by 'caret'.

Value

The population total estimation (or mean if specified by the ‘estimate_mean‘ parameter).

References

Valliant, R., Dorfman, A. H., & Royall, R. M. (2000) *Finite population sampling and inference: a prediction approach.* Wiley, New York.

Examples

```
#Simple example
covariates = c("education_primaria", "education_secundaria",
   "age", "sex", "language")
if (is.numeric(sampleNP$vote_gen))
   sampleNP$vote_gen = factor(sampleNP$vote_gen, c(0, 1), c('F', 'T'))
model_based(sampleNP, population, covariates,
   "vote_gen", positive_label = 'T', algorithm = 'glmnet')
```

`model_calibrated` *Calculates a model calibrated estimation*

Description

It uses the model calibrated estimator introduced by Wu et al. (2001).

Usage

```
model_calibrated(
  sample_data,
  weights,
  full_data,
  covariates,
  estimated_var,
  estimate_mean = FALSE,
  positive_label = NULL,
  algorithm = "glm",
  proc = NULL,
  ...
)
```

Arguments

`sample_data`	Data frame containing the sample.
`weights`	Vector containing the sample weights.
`full_data`	Data frame containing all the individuals contained in the population.
`covariates`	String vector specifying the common variables to use for training.

`estimated_var`	String specifying the variable to estimate.
`estimate_mean`	Boolean specifying whether the mean estimation should be returned. Otherwise, the total estimation is returned by default.
`positive_label`	String specifying the label to be considered positive if the estimated variable is categorical. Leave it as the default NULL otherwise.
`algorithm`	A string specifying which classification or regression model to use (same as caret's method).
`proc`	A string or vector of strings specifying if any of the data preprocessing techniques available in train function from 'caret' package should be applied to data prior to the propensity estimation. By default, its value is NULL and no preprocessing is applied.
`...`	Further parameters to be passed to the train function.

Details

Training of the models is done via the 'caret' package. The algorithm specified in `algorithm` must match one of the names in the list of algorithms supported by 'caret'.

Value

The population total estimation (or mean if specified by the 'estimate_mean' parameter).

References

Wu, C., & Sitter, R. R. (2001). *A model-calibration approach to using complete auxiliary information from survey data.* Journal of the American Statistical Association, 96(453), 185-193.

Examples

```
#Simple example
covariates = c("education_primaria", "education_secundaria",
   "age", "sex", "language")
if (is.numeric(sampleNP$vote_gen))
   sampleNP$vote_gen = factor(sampleNP$vote_gen, c(0, 1), c('F', 'T'))
model_calibrated(sampleNP, nrow(population) / nrow(sampleNP),
   population, covariates, "vote_gen", positive_label = 'T', algorithm = 'glmnet')
```

`population` *A full population*

Description

A dataset of a simulated fictitious population of 50,000 individuals. Further details on the generation of the dataset can be found in Ferri-García and Rueda (2018). The variables present in the dataset are the following:

- education_primaria. A binary variable indicating if the highest academic level achieved by the individual is Primary Education.
- education_secundaria. A binary variable indicating if the highest academic level achieved by the individual is Secondary Education.
- education_terciaria. A binary variable indicating if the highest academic level achieved by the individual is Tertiary Education.
- age. A numeric variable, with values ranging from 18 to 100, indicating the age of the individual.
- sex. A binary variable indicating if the individual is a man.
- language. A binary variable indicating if the individual is a native.

Usage

```
population
```

Format

An object of class `data.frame` with 50000 rows and 6 columns.

References

Ferri-García, R., & Rueda, M. (2018). *Efficiency of propensity score adjustment and calibration on the estimation from non-probabilistic online surveys*. SORT-Statistics and Operations Research Transactions, 1(2), 159-162.

`propensities` *Calculates sample propensities*

Description

Given a convenience sample and a reference sample, computes estimates on the propensity to participate in the convenience sample based on classification models to be selected by the user.

Usage

```
propensities(
  convenience_sample,
  reference_sample,
  covariates,
  algorithm = "glm",
  smooth = FALSE,
  proc = NULL,
  trControl = trainControl(classProbs = TRUE),
  ...
)
```

Arguments

`convenience_sample`	Data frame containing the non-probabilistic sample.
`reference_sample`	Data frame containing the probabilistic sample.
`covariates`	String vector specifying the common variables to use for training.
`algorithm`	A string specifying which classification or regression model to use (same as caret's method).
`smooth`	A logical value; if TRUE, propensity estimates pi_i are smoothed applying the formula (1000*pi_i + 0.5)/1001
`proc`	A string or vector of strings specifying if any of the data preprocessing techniques available in train function from ‘caret‘ package should be applied to data prior to the propensity estimation. By default, its value is NULL and no preprocessing is applied.
`trControl`	A trainControl specifying the computational nuances of the train function.
`...`	Further parameters to be passed to the train function.

Details

Training of the propensity estimation models is done via the ‘caret‘ package. The algorithm specified in `algorithm` must match one of the names in the list of algorithms supported by ‘caret‘. Case weights are used to balance classes (for models that accept them). The smoothing formula for propensities avoids mathematical irregularities in the calculation of sample weight when an estimated propensity is 0 or 1. Further details can be found in Buskirk and Kolenikov (2015).

Value

A list containing ‘convenience‘ propensities and ‘reference‘ propensities.

References

Buskirk, T. D., & Kolenikov, S. (2015). *Finding respondents in the forest: A comparison of logistic regression and random forest models for response propensity weighting and stratification.* Survey Methods: Insights from the Field, 17.

Examples

```
#Simple example with default parameters
covariates = c("education_primaria", "education_secundaria")
propensities(sampleNP, sampleP, covariates)
```

prop_estimation *Estimates the population proportion*

Description

Estimates the proportion of a given class or classes for the specified variables measured in a sample given some pre-calculated weights.

Usage

```
prop_estimation(sample, weights, estimated_vars, class, N = NULL)
```

Arguments

`sample`	A data frame containing the sample with the variables for which the means are to be calculated.
`weights`	A vector of pre-calculated weights.
`estimated_vars`	String vector specifying the variables in the sample to be estimated.
`class`	String vector specifying which class (value) proportion is to be estimated in each variable. The *i*-th element of this vector corresponds to the class of which proportion is desired to estimate of the *i*-th variable of the vector specified in `estimated_vars`.
`N`	An integer specifying the population size (optional).

Value

A vector with the corresponding estimations.

Examples

```
covariates = c("education_primaria", "education_secundaria")
data_propensities = propensities(sampleNP, sampleP, covariates)
psa_weights = sc_weights(data_propensities$convenience)

#The function will estimate the proportion of individuals
#with the 0 value in vote_pens and the 1 value in vote_pir
prop_estimation(sampleNP, psa_weights, c("vote_pens", "vote_pir"), c(0, 1))
```

`sampleNP` *A non-probabilistic sample*

Description

A dataset of 1000 individuals extracted from the subpopulation of individuals with internet access in a simulated fictitious population of 50,000 individuals. This sample attempts to reproduce a case of nonprobability sampling with selection bias, as there are important differences between the potentially covered population, the covered population and the full target population. Further details on the generation of the dataset can be found in Ferri-García and Rueda (2018). The variables present in the dataset are the following:

- vote_gen. A binary variable indicating if the individual vote preferences are for Party 1. This variable is related to gender.
- vote_pens. A binary variable indicating if the individual vote preferences are for Party 2. This variable is related to age.
- vote_pir. A binary variable indicating if the individual vote preferences are for Party 3. This variable is related to age and internet access.
- education_primaria. A binary variable indicating if the highest academic level achieved by the individual is Primary Education.
- education_secundaria. A binary variable indicating if the highest academic level achieved by the individual is Secondary Education.
- education_terciaria. A binary variable indicating if the highest academic level achieved by the individual is Tertiary Education.
- age. A numeric variable, with values ranging from 18 to 100, indicating the age of the individual.
- sex. A binary variable indicating if the individual is a man.
- language. A binary variable indicating if the individual is a native.

Usage

```
sampleNP
```

Format

An object of class `data.frame` with 1000 rows and 9 columns.

References

Ferri-García, R., & Rueda, M. (2018). *Efficiency of propensity score adjustment and calibration on the estimation from non-probabilistic online surveys*. SORT-Statistics and Operations Research Transactions, 1(2), 159-162.

sampleP *A probabilistic sample*

Description

A dataset of 500 individuals extracted with simple random sampling from a simulated fictitious population of 50,000 individuals. Further details on the generation of the dataset can be found in Ferri-García and Rueda (2018). The variables present in the dataset are the following:

- education_primaria. A binary variable indicating if the highest academic level achieved by the individual is Primary Education.
- education_secundaria. A binary variable indicating if the highest academic level achieved by the individual is Secondary Education.
- education_terciaria. A binary variable indicating if the highest academic level achieved by the individual is Tertiary Education.
- age. A numeric variable, with values ranging from 18 to 100, indicating the age of the individual.
- sex. A binary variable indicating if the individual is a man.

Usage

```
sampleP
```

Format

An object of class `data.frame` with 500 rows and 5 columns.

References

Ferri-García, R., & Rueda, M. (2018). *Efficiency of propensity score adjustment and calibration on the estimation from non-probabilistic online surveys*. SORT-Statistics and Operations Research Transactions, 1(2), 159-162.

sc_weights *Calculates Schonlau and Couper weights*

Description

Computes weights from propensity estimates using the (1 - pi_i)/pi_i formula introduced in Schonlau and Couper (2017).

Usage

```
sc_weights(propensities)
```

Arguments

`propensities`	A vector with the propensities associated to the elements of the convenience sample.

Details

The function takes the vector of propensities $\pi(x)$ and calculates the weights to be applied in the Hajek estimator using the formula that can be found in Schonlau and Couper (2017). For an individual *i*, weight is calculated as follows:

$$w_i = \frac{1 - \pi_i(x)}{\pi_i(x)}$$

Value

A vector with the corresponding weights.

References

Schonlau, M., & Couper, M. P. (2017). *Options for conducting web surveys.* Statistical Science, 32(2), 279-292.

Examples

```
covariates = c("education_primaria", "education_secundaria")
data_propensities = propensities(sampleNP, sampleP, covariates)
sc_weights(data_propensities$convenience)
```

`total_estimation` *Estimates the population totals*

Description

Estimates the population totals for the specified variables measured in a sample given some pre-calculated weights.

Usage

```
total_estimation(sample, weights, estimated_vars, N)
```

Arguments

`sample`	A data frame containing the sample with the variables for which the estimated population totals are to be calculated.
`weights`	A vector of pre-calculated weights.
`estimated_vars`	String vector specifying the variables in the sample to be estimated.
`N`	An integer specifying the population size.

Value

A vector with the corresponding estimations.

Examples

```
covariates = c("education_primaria", "education_secundaria")
data_propensities = propensities(sampleNP, sampleP, covariates)
psa_weights = sc_weights(data_propensities$convenience)
total_estimation(sampleNP, psa_weights, c("vote_pens"), 50000)
```

`valliant_weights` *Calculates Valliant weights*

Description

Computes weights from propensity estimates using the 1/pi_i formula introduced in Valliant (2019).

Usage

```
valliant_weights(propensities)
```

Arguments

`propensities`	A vector with the propensities associated to the elements of the convenience sample.

Details

The function takes the vector of propensities $\pi(x)$ and calculates the weights to be applied in the Hajek estimator using the formula that can be found in Valliant (2019). For an individual *i*, weight is calculated as follows:

$$w_i = 1/\pi_i(x)$$

Value

A vector with the corresponding weights.

References

Valliant, R. (2019). *Comparing Alternatives for Estimation from Nonprobability Samples*. Journal of Survey Statistics and Methodology, smz003, https://doi.org/10.1093/jssam/smz003

Examples

```
covariates = c("education_primaria", "education_secundaria")
data_propensities = propensities(sampleNP, sampleP, covariates)
valliant_weights(data_propensities$convenience)
```

`vd_weights` *Calculates Valliant and Dever weights*

Description

Computes weights from propensity estimates using the propensity stratification 1/p_i averaging formula introduced in Valliant and Dever (2011).

Usage

```
vd_weights(convenience_propensities, reference_propensities, g = 5)
```

Arguments

`convenience_propensities`	A vector with the propensities associated with the convenience sample.
`reference_propensities`	A vector with the propensities associated with the reference sample.
`g`	The number of strata to use; by default, its value is 5.

Details

The function takes the vector of propensities $\pi(x)$ and calculates the weights to be applied in the Horvitz-Thompson estimator using the formula that can be found in Valliant and Dever (2019). The vector of propensities is divided in *g* strata (ideally five according to Cochran, 1968) aiming to have individuals with similar propensities in each strata. After the stratification, weight is calculated as follows for an individual *i*:

$$w_i = \frac{n(g_i)}{\sum_{k \in g_i} \pi_k(x)}$$

where g_i represents the strata to which *i* belongs, and $n(g_i)$ is the number of individuals in the g_i strata.

Value

A vector with the corresponding weights.

References

Valliant, R., & Dever, J. A. (2011). *Estimating propensity adjustments for volunteer web surveys.* Sociological Methods & Research, 40(1), 105-137.

Cochran, W. G. (1968). *The Effectiveness of Adjustment by Subclassification in Removing Bias in Observational Studies.* Biometrics, 24(2), 295-313

Examples

```
covariates = c("education_primaria", "education_secundaria")
data_propensities = propensities(sampleNP, sampleP, covariates)
vd_weights(data_propensities$convenience, data_propensities$reference)
```

Index