Package 'nonprobsvy'

November 14, 2024

Type Package

Title Inference Based on Non-Probability Samples

Version 0.1.1

Description Statistical inference with non-probability samples when auxiliary information from external sources such as probability samples or population totals or means is available. Details can be found in: Wu et al. (2020) <doi:10.1080/01621459.2019.1677241>, Kim et al. (2021) <doi:10.1111/rssa.12696>, Kim et al. (2021) <https://www150.statcan.gc.ca/n1/pub/12-001-x/2022002/article/00002-eng.htm>, Kim et al. (2021) <https://www150.statcan.gc.ca/n1/pub/12-001-x/2021001/article/00004-eng.htm>, Kim et al. (2020) <doi:10.1111/rssb.12354>.

License MIT + file LICENSE

Encoding UTF-8

RdMacros mathjaxr

RoxygenNote 7.3.2

URL https://github.com/ncn-foreigners/nonprobsvy, https://ncn-foreigners.github.io/nonprobsvy/

BugReports https://github.com/ncn-foreigners/nonprobsvy/issues

Suggests tinytest, covr, sampling, spelling

Depends R (>= 4.0.0), survey

Imports maxLik, stats, Matrix, MASS, ncvreg, mathjaxr, RANN, Rcpp (>= 1.0.12), nleqslv, doParallel, foreach, parallel

LinkingTo Rcpp, RcppArmadillo

Language en-US

NeedsCompilation yes

Author Łukasz Chrostowski [aut, cre], Maciej Beręsewicz [aut, ctb] (<https://orcid.org/0000-0002-8281-4301>), Piotr Chlebicki [aut, ctb]

Maintainer Łukasz Chrostowski <lukchr@st.amu.edu.pl>

Repository CRAN

Date/Publication 2024-11-14 07:20:02 UTC

Contents

cloglog_model_nonprobsvy 2
confint.nonprobsvy 3
controlInf 3
controlOut 4
controlSel 6
genSimData 8
logit_model_nonprobsvy 9
nonprob 10
pop.size 18
probit_model_nonprobsvy 19
summary.nonprobsvy 19
vcov.nonprobsvy 21

Index **22**

`cloglog_model_nonprobsvy` *Complementary log-log model for weights adjustment*

Description

`cloglog_model_nonprobsvy` returns all the methods/objects/functions required to estimate the model, assuming a cloglog link function.

Usage

```
cloglog_model_nonprobsvy(...)
```

Arguments

...	Additional, optional arguments.

Value

List with selected methods/objects/functions.

Author(s)

Łukasz Chrostowski, Maciej Beręsewicz

See Also

`nonprob()` – for fitting procedure with non-probability samples.

confint.nonprobsvy *Confidence Intervals for Model Parameters*

Description

A function that computes confidence intervals for selection model coefficients.

Usage

```
## S3 method for class 'nonprobsvy'
confint(object, parm, level = 0.95, ...)
```

Arguments

object	object of nonprobsvy class.
parm	names of parameters for which confidence intervals are to be computed, if missing all parameters will be considered.
level	confidence level for intervals.
...	additional arguments

Value

An object with named columns that include upper and lower limit of confidence intervals.

controlInf *Control parameters for inference*

Description

controlInf constructs a list with all necessary control parameters for statistical inference.

Usage

```
controlInf(
  vars_selection = FALSE,
  var_method = c("analytic", "bootstrap"),
 rep_type = c("auto", "JK1", "JKn", "BRR", "bootstrap", "subbootstrap", "mrbbootstrap",
    "Fay"),
  bias_inf = c("union", "div"),
  num_boot = 500,
  bias_correction = FALSE,
  alpha = 0.05,
  cores = 1,
  keep_boot,
  nn_exact_se = FALSE,
  pi_ij
)
```

Arguments

vars_selection	If TRUE, then variables selection model is used.
var_method	variance method.
rep_type	replication type for weights in the bootstrap method for variance estimation passed to survey::as.svrepdesign(). Default is subbootstrap.
bias_inf	inference method in the bias minimization. • if union then final model is fitting on union of selected variables for selection and outcome models • if div then final model is fitting separately on division of selected variables into relevant ones for selection and outcome model.
num_boot	number of iteration for bootstrap algorithms.
bias_correction	if TRUE, then bias minimization estimation used during fitting the model.
alpha	Significance level, Default is 0.05.
cores	Number of cores in parallel computing.
keep_boot	Logical indicating whether statistics from bootstrap should be kept. By default set to TRUE
nn_exact_se	Logical value indicating whether to compute the exact standard error estimate for nn or pmm estimator. The variance estimator for estimation based on nn or pmm can be decomposed into three parts, with the third being computed using covariance between imputed values for units in probability sample using predictive matches from non-probability sample. In most situations this term is negligible and is very computationally expensive so by default this is set to FALSE, but it is recommended to set this value to TRUE before submitting final results.
pi_ij	TODO, either matrix or ppsmat class object.

Value

List with selected parameters.

See Also

nonprob() – for fitting procedure with non-probability samples.

---

controlOut *Control parameters for outcome model*

Description

controlOut constructs a list with all necessary control parameters for outcome model.

Usage

```
controlOut(
  epsilon = 1e-04,
  maxit = 100,
  trace = FALSE,
  k = 1,
  penalty = c("SCAD", "lasso", "MCP"),
  a_SCAD = 3.7,
  a_MCP = 3,
  lambda_min = 0.001,
  nlambda = 100,
  nfolds = 10,
  treetype = "kd",
  searchtype = "standard",
  predictive_match = 1:2,
  pmm_weights = c("none", "prop_dist"),
  pmm_k_choice = c("none", "min_var"),
  pmm_reg_engine = c("glm", "loess")
)
```

Arguments

epsilon	Tolerance for fitting algorithms. Default is 1e-6.
maxit	Maximum number of iterations.
trace	logical value. If TRUE trace steps of the fitting algorithms. Default is FALSE.
k	The k parameter in the RANN::nn2() function. Default is 5.
penalty	penalty algorithm for variable selection. Default is SCAD
a_SCAD	The tuning parameter of the SCAD penalty for outcome model. Default is 3.7.
a_MCP	The tuning parameter of the MCP penalty for outcome model. Default is 3.
lambda_min	The smallest value for lambda, as a fraction of lambda.max. Default is .001.
nlambda	The number of lambda values. Default is 100.
nfolds	The number of folds during cross-validation for variables selection model.
treetype	Type of tree for nearest neighbour imputation passed to RANN::nn2() function.
searchtype	Type of search for nearest neighbour imputation passed to RANN::nn2() function.
predictive_match	(Only for predictive mean matching) Indicates how to select 'closest' unit from nonprobability sample for each unit in probability sample. Either 1 (default) or 2 where 2 is matching by minimizing distance between $\hat{y}_i$ for $i \in S_A$ and y_j for $j \in S_B$ and 1 is matching by minimizing distance between $\hat{y}_i$ for $i \in S_A$ and $\hat{y}_i$ for $i \in S_A$.
pmm_weights	(Only for predictive mean matching) Indicate how to weight k nearest neighbours in S_B to create imputed value for units in S_A. The default value "none" indicates that mean of k nearest y's from S_B should be used whereas "prop_dist" results in weighted mean of these k values where weights are inversely proportional to distance between matched values.

pmm_k_choice	Character value indicating how k hyper-parameter should be chosen, by default "none" meaning k provided in control_outcome argument will be used. For now the only other option "min_var" means that k will be chosen by minimizing estimated variance of estimator for mean. Parameter k provided in this control list will be chosen as starting point.
pmm_reg_engine	TODO

Value

List with selected parameters.

See Also

nonprob() – for fitting procedure with non-probability samples.

---

controlSel *Control parameters for selection model*

Description

controlSel constructs a list with all necessary control parameters for selection model.

Usage

```
controlSel(
  method = "glm.fit",
  epsilon = 1e-04,
  maxit = 500,
  trace = FALSE,
  optimizer = c("maxLik", "optim"),
  maxLik_method = "NR",
  optim_method = "BFGS",
  dependence = FALSE,
  key = NULL,
  est_method_sel = c("mle", "gee"),
  h = c(1, 2),
  penalty = c("SCAD", "lasso", "MCP"),
  a_SCAD = 3.7,
  a_MCP = 3,
  lambda = -1,
  lambda_min = 0.001,
  nlambda = 50,
  nfolds = 10,
  print_level = 0,
  start_type = c("glm", "naive", "zero"),
  nleqslv_method = c("Broyden", "Newton"),
 nleqslv_global = c("dbldog", "pwldog", "cline", "qline", "gline", "hook", "none"),
  nleqslv_xscalm = c("fixed", "auto")
)
```

Arguments

method	estimation method.
epsilon	Tolerance for fitting algorithms by default 1e-6.
maxit	Maximum number of iterations.
trace	logical value. If TRUE trace steps of the fitting algorithms. Default is FALSE
optimizer	• optimization function for maximum likelihood estimation.
maxLik_method	maximisation method that will be passed to maxLik::maxLik() function. Default is NR.
optim_method	maximisation method that will be passed to stats::optim() function. Default is BFGS.
dependence	logical value - TRUE if samples are dependent.
key	binary key variable
est_method_sel	Method of estimation for propensity score model.
h	Smooth function for the generalized estimating equations methods taking the following values • if 1 then $\mathbf{h}(\mathbf{x}, \boldsymbol{\theta}) = \frac{\pi(\mathbf{x}, \boldsymbol{\theta})}{\mathbf{x}}$ • if 2 then $\mathbf{h}(\mathbf{x}, \boldsymbol{\theta}) = \mathbf{x}$
penalty	The penanlization function used during variables selection.
a_SCAD	The tuning parameter of the SCAD penalty for selection model. Default is 3.7.
a_MCP	The tuning parameter of the MCP penalty for selection model. Default is 3.
lambda	A user-specified λ value during variable selection model fitting.
lambda_min	The smallest value for lambda, as a fraction of lambda.max. Default is .001.
nlambda	The number of lambda values. Default is 50.
nfolds	The number of folds for cross validation. Default is 10.
print_level	this argument determines the level of printing which is done during the optimization (for propensity score model) process.
start_type	• Type of method for start points for model fitting taking the following values – if glm then start taken from the glm function called on samples. – if naive then start consists of a vector which has the value of an estimated parameter for one-dimensional data (on intercept) and 0 for the rest. – if zero then start is a vector of zeros.
nleqslv_method	The method that will be passed to nleqslv::nleqslv() function.
nleqslv_global	The global strategy that will be passed to nleqslv::nleqslv() function.
nleqslv_xscalm	The type of x scaling that will be passed to nleqslv::nleqslv() function.

Value

List with selected parameters.

Author(s)

Łukasz Chrostowski, Maciej Beręsewicz

See Also

`nonprob()` – for fitting procedure with non-probability samples.

---

genSimData *Simulation data*

Description

Generate simulated data according to Chen, Li & Wu (2020), section 5.

Usage

```
genSimData(N = 10000, n = 1000)
```

Arguments

N	integer, population size, default 10000
n	integer, big data sample, default 1000

Value

genSimData returns a data.frame, with the following columns:

- x0 – intercept
- x1 – the first variable based on z1
- x2 – the second variable based on z2 and x1
- x3 – the third variable based on z3 and x1 and x2
- x4 – the third variable based on z4 and x1, x2 and x3
- $y30$ – y generated from the model $y = 2 + x1 + x2 + x3 + x4 + \sigma \cdot \varepsilon$, so the cor(y,y_hat) = 0.30
- $y60$ – y generated from the model $y = 2 + x1 + x2 + x3 + x4 + \sigma \cdot \varepsilon$, so the cor(y,y_hat) = 0.60
- $y80$ – y generated from the model $y = 2 + x1 + x2 + x3 + x4 + \sigma \cdot \varepsilon$, so the cor(y,y_hat) = 0.80
- rho – true propensity scores for big data such that sum(rho)=n
- srs – probabilities of inclusion to random sample such that max(srs)/min(srs)=50

Author(s)

Łukasz Chrostowski, Maciej Beręsewicz

References

Chen, Y., Li, P., & Wu, C. (2020). Doubly Robust Inference With Nonprobability Survey Samples. Journal of the American Statistical Association, 115(532), 2011–2021. doi:10.1080/01621459.2019.1677241

Examples

```
## generate data with N=20000 and n=2000
genSimData(N = 20000, n = 2000)

## generate data when big data is almost as N
genSimData(N = 10000, n = 9000)
```

`logit_model_nonprobsvy`

Logit model for weights adjustment

Description

`logit_model_nonprobsvy` returns all the methods/objects/functions required to estimate the model, assuming a logit link function.

Usage

```
logit_model_nonprobsvy(...)
```

Arguments

...	Additional, optional arguments.

Value

List with selected methods/objects/functions.

Author(s)

Łukasz Chrostowski, Maciej Beręsewicz

See Also

`nonprob()` – for fitting procedure with non-probability samples.

nonprob *Inference with the non-probability survey samples*

Description

nonprob fits model for inference based on non-probability surveys (including big data) using various methods. The function allows you to estimate the population mean with access to a reference probability sample, as well as sums and means of covariates.

The package implements state-of-the-art approaches recently proposed in the literature: Chen et al. (2020), Yang et al. (2020), Wu (2022) and uses the Lumley 2004 survey package for inference.

It provides propensity score weighting (e.g. with calibration constraints), mass imputation (e.g. nearest neighbour) and doubly robust estimators that take into account minimisation of the asymptotic bias of the population mean estimators or variable selection. The package uses survey package functionality when a probability sample is available.

Usage

```
nonprob(
  data,
  selection = NULL,
  outcome = NULL,
  target = NULL,
  svydesign = NULL,
  pop_totals = NULL,
  pop_means = NULL,
  pop_size = NULL,
  method_selection = c("logit", "cloglog", "probit"),
  method_outcome = c("glm", "nn", "pmm"),
  family_outcome = c("gaussian", "binomial", "poisson"),
  subset = NULL,
  strata = NULL,
  weights = NULL,
  na_action = NULL,
  control_selection = controlSel(),
  control_outcome = controlOut(),
  control_inference = controlInf(),
  start_selection = NULL,
  start_outcome = NULL,
  verbose = FALSE,
  x = TRUE,
  y = TRUE,
  se = TRUE,
  ...
)
```

Arguments

data	data.frame with data from the non-probability sample.
selection	formula, the selection (propensity) equation.
outcome	formula, the outcome equation.
target	formula with target variables.
svydesign	an optional svydesign object (from the survey package) containing probability sample and design weights.
pop_totals	an optional named vector with population totals of the covariates.
pop_means	an optional named vector with population means of the covariates.
pop_size	an optional double with population size.
method_selection	a character with method for propensity scores estimation.
method_outcome	a character with method for response variable estimation.
family_outcome	a character string describing the error distribution and link function to be used in the model. Default is "gaussian". Currently supports: gaussian with identity link, poisson and binomial.
subset	an optional vector specifying a subset of observations to be used in the fitting process.
strata	an optional vector specifying strata.
weights	an optional vector of prior weights to be used in the fitting process. Should be NULL or a numeric vector. It is assumed that this vector contains frequency or analytic weights.
na_action	a function which indicates what should happen when the data contain NAs.
control_selection	a list indicating parameters to use in fitting selection model for propensity scores.
control_outcome	a list indicating parameters to use in fitting model for outcome variable.
control_inference	a list indicating parameters to use in inference based on probability and non-probability samples, contains parameters such as estimation method or variance method.
start_selection	an optional vector with starting values for the parameters of the selection equation.
start_outcome	an optional vector with starting values for the parameters of the outcome equation.
verbose	verbose, numeric.
x	Logical value indicating whether to return model matrix of covariates as a part of output.
y	Logical value indicating whether to return vector of outcome variable as a part of output.
se	Logical value indicating whether to calculate and return standard error of estimated mean.
...	Additional, optional arguments.

Details

Let y be the response variable for which we want to estimate the population mean, given by

$$\mu_y = \frac{1}{N}\sum_{i=1}^{N} y_i.$$

For this purpose we consider data integration with the following structure. Let S_A be the non-probability sample with the design matrix of covariates as

$$\boldsymbol{X}_A = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n_A 1} & x_{n_A 2} & \cdots & x_{n_A p} \end{bmatrix}$$

and vector of outcome variable

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n_A}. \end{bmatrix}$$

On the other hand, let S_B be the probability sample with design matrix of covariates be

$$\boldsymbol{X}_B = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n_B 1} & x_{n_B 2} & \cdots & x_{n_B p}. \end{bmatrix}$$

Instead of a sample of units we can consider a vector of population sums in the form of $\tau_x = (\sum_{i\in\mathcal{U}} \boldsymbol{x}_{i1}, \sum_{i\in\mathcal{U}} \boldsymbol{x}_{i2}, ..., \sum_{i\in\mathcal{U}} \boldsymbol{x}_{ip})$ or means $\frac{\tau_x}{N}$, where $\mathcal{U}$ refers to a finite population. Note that we do not assume access to the response variable for S_B. In general we make the following assumptions:

1. The selection indicator of belonging to non-probability sample R_i and the response variable y_i are independent given the set of covariates $\boldsymbol{x}_i$.
2. All units have a non-zero propensity score, i.e., $\pi_i^A > 0$ for all i.
3. The indicator variables R_i^A and R_j^A are independent for given $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ for $i \neq j$.

There are three possible approaches to the problem of estimating population mean using non-probability samples:

1. Inverse probability weighting - The main drawback of non-probability sampling is the unknown selection mechanism for a unit to be included in the sample. This is why we talk about the so-called "biased sample" problem. The inverse probability approach is based on the assumption that a reference probability sample is available and therefore we can estimate the propensity score of the selection mechanism. The estimator has the following form:

$$\hat{\mu}_{IPW} = \frac{1}{N^A}\sum_{i\in S_A} \frac{y_i}{\hat{\pi}_i^A}.$$

For this purpose several estimation methods can be considered. The first approach is maximum likelihood estimation with a corrected log-likelihood function, which is given by the following formula

$$\ell^*(\boldsymbol{\theta}) = \sum_{i \in S_A} \log \left\{ \frac{\pi(\boldsymbol{x}_i, \boldsymbol{\theta})}{1 - \pi(\boldsymbol{x}_i, \boldsymbol{\theta})} \right\} + \sum_{i \in S_B} d_i^B \log \left\{ 1 - \pi(\boldsymbol{x}_i, \boldsymbol{\theta}) \right\}.$$

In the literature, the main approach to modelling propensity scores is based on the logit link function. However, we extend the propensity score model with the additional link functions such as cloglog and probit. The pseudo-score equations derived from ML methods can be replaced by the idea of generalised estimating equations with calibration constraints defined by equations.

$$\mathbf{U}(\boldsymbol{\theta}) = \sum_{i \in S_A} \mathbf{h}\left(\mathbf{x}_i, \boldsymbol{\theta}\right) - \sum_{i \in S_B} d_i^B \pi\left(\mathbf{x}_i, \boldsymbol{\theta}\right) \mathbf{h}\left(\mathbf{x}_i, \boldsymbol{\theta}\right).$$

Notice that for $\mathbf{h}\left(\mathbf{x}_i, \boldsymbol{\theta}\right) = \frac{\pi(\boldsymbol{x}, \boldsymbol{\theta})}{\boldsymbol{x}}$ We do not need a probability sample and can use a vector of population totals/means.

2. Mass imputation – This method is based on a framework where imputed values of outcome variables are created for the entire probability sample. In this case, we treat the large sample as a training data set that is used to build an imputation model. Using the imputed values for the probability sample and the (known) design weights, we can build a population mean estimator of the form:

$$\hat{\mu}_{MI} = \frac{1}{N^B} \sum_{i \in S_B} d_i^B \hat{y}_i.$$

It opens the the door to a very flexible method for imputation models. The package uses generalized linear models from `stats::glm()`, the nearest neighbour algorithm using `RANN::nn2()` and predictive mean matching.

3. Doubly robust estimation – The IPW and MI estimators are sensitive to misspecified models for the propensity score and outcome variable, respectively. To this end, so-called doubly robust methods are presented that take these problems into account. It is a simple idea to combine propensity score and imputation models during inference, leading to the following estimator

$$\hat{\mu}_{DR} = \frac{1}{N^A} \sum_{i \in S_A} \hat{d}_i^A (y_i - \hat{y}_i) + \frac{1}{N^B} \sum_{i \in S_B} d_i^B \hat{y}_i.$$

In addition, an approach based directly on bias minimisation has been implemented. The following formula

$$\begin{aligned} bias(\hat{\mu}_{DR}) =& \mathbb{E}(\hat{\mu}_{DR} - \mu) \\ =& \mathbb{E}\left\{ \frac{1}{N} \sum_{i=1}^{N} (\frac{R_i^A}{\pi_i^A(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\theta})} - 1)(y_i - \mathrm{m}(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta})) \right\} \\ &+ \mathbb{E}\left\{ \frac{1}{N} \sum_{i=1}^{N} (R_i^B d_i^B - 1)\, \mathrm{m}(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta}) \right\}, \end{aligned}$$

lead us to system of equations

$$J(\theta, \beta) = \left\{ \begin{array}{c} J_1(\theta, \beta) \\ J_2(\theta, \beta) \end{array} \right\} = \left\{ \begin{array}{c} \sum_{i=1}^{N} R_i^A \left\{ \frac{1}{\pi(\boldsymbol{x}_i, \boldsymbol{\theta})} - 1 \right\} \{ y_i - m(\boldsymbol{x}_i, \boldsymbol{\beta}) \}\, \boldsymbol{x}_i \\ \sum_{i=1}^{N} \frac{R_i^A}{\pi(\boldsymbol{x}_i, \boldsymbol{\theta})} \frac{\partial m(\boldsymbol{x}_i, \boldsymbol{\beta})}{\partial \boldsymbol{\beta}} - \sum_{i \in \mathcal{S}_\mathrm{B}} d_i^\mathrm{B} \frac{\partial m(\boldsymbol{x}_i, \boldsymbol{\beta})}{\partial \boldsymbol{\beta}} \end{array} \right\},$$

where $m(\boldsymbol{x}_i, \boldsymbol{\beta})$ is a mass imputation (regression) model for the outcome variable and propensity scores π_i^A are estimated using a `logit` function for the model. As with the `MLE` and `GEE` approaches we have extended this method to `cloglog` and `probit` links.

As it is not straightforward to calculate the variances of these estimators, asymptotic equivalents of the variances derived using the Taylor approximation have been proposed in the literature. Details can be found here. In addition, a bootstrap approach can be used for variance estimation.

The function also allows variables selection using known methods that have been implemented to handle the integration of probability and non-probability sampling. In the presence of high-dimensional data, variable selection is important, because it can reduce the variability in the estimate that results from using irrelevant variables to build the model. Let $\mathrm{U}(\boldsymbol{\theta}, \boldsymbol{\beta})$ be the joint estimating function for $(\boldsymbol{\theta}, \boldsymbol{\beta})$. We define the penalized estimating functions as

$$\mathrm{U}^p(\boldsymbol{\theta}, \boldsymbol{\beta}) = \mathrm{U}(\boldsymbol{\theta}, \boldsymbol{\beta}) - \left\{ \; q_{\lambda_\theta}(|\boldsymbol{\theta}|)\,\mathrm{sgn}(\boldsymbol{\theta})\; q_{\lambda_\beta}(|\boldsymbol{\beta}|)\,\mathrm{sgn}(\boldsymbol{\beta}) \; \right\},$$

where λ_θ and q_{λ_β} are some smooth functions. We let $q_\lambda(x) = \frac{\partial p_\lambda}{\partial x}$, where p_λ is some penalization function. Details of penalization functions and techniques for solving this type of equation can be found here. To use the variable selection model, set the `vars_selection` parameter in the `controlInf()` function to `TRUE`. In addition, in the other control functions such as `controlSel()` and `controlOut()` you can set parameters for the selection of the relevant variables, such as the number of folds during cross-validation algorithm or the lambda value for penalizations. Details can be found in the documentation of the control functions for `nonprob`.

Value

Returns an object of class `c("nonprobsvy", "nonprobsvy_dr")` in case of doubly robust estimator, `c("nonprobsvy", "nonprobsvy_mi")` in case of mass imputation estimator and `c("nonprobsvy", "nonprobsvy_ipw")` in case of inverse probability weighting estimator with type `list` containing:

- `X` – model matrix containing data from probability and non-probability samples if specified at a function call.
- `y` – list of vector of outcome variables if specified at a function call.
- `R` – vector indicating the probablistic (0) or non-probablistic (1) units in the matrix X.
- `prob` – vector of estimated propensity scores for non-probability sample.
- `weights` – vector of estimated weights for non-probability sample.
- `control` – list of control functions.
- `output` – output of the model with information on the estimated population mean and standard errors.
- `SE` – standard error of the estimator of the population mean, divided into errors from probability and non-probability samples.
- `confidence_interval` – confidence interval of population mean estimator.
- `nonprob_size` – size of non-probability sample.
- `prob_size` – size of probability sample.
- `pop_size` – estimated population size derived from estimated weights (non-probability sample) or known design weights (probability sample).

- pop_totals – the total values of the auxiliary variables derived from a probability sample or vector of total/mean values.
- outcome – list containing information about the fitting of the mass imputation model, in the case of regression model the object containing the list returned by stats::glm(), in the case of the nearest neighbour imputation the object containing list returned by RANN::nn2(). If bias_correction in controlInf() is set to TRUE, the estimation is based on the joint estimating equations for the selection and outcome model and therefore, the list is different from the one returned by the stats::glm() function and contains elements such as
  - coefficients – estimated coefficients of the regression model.
  - std_err – standard errors of the estimated coefficients.
  - residuals – The response residuals.
  - variance_covariance – The variance-covariance matrix of the coefficient estimates.
  - df_residual – The degrees of freedom for residuals.
  - family – specifies the error distribution and link function to be used in the model.
  - fitted.values – The predicted values of the response variable based on the fitted model.
  - linear.predictors – The linear fit on link scale.
  - X – The design matrix.
  - method – set on glm, since the regression method.
  - model_frame – Matrix of data from probability sample used for mass imputation.

  In addition, if the variable selection model for the outcome variable is fitting, the list includes the
  - cve – the error for each value of lambda, averaged across the cross-validation folds.
- selection – list containing information about fitting of propensity score model, such as
  - coefficients – a named vector of coefficients.
  - std_err – standard errors of the estimated model coefficients.
  - residuals – the response residuals.
  - variance – the root mean square error.
  - fitted_values – the fitted mean values, obtained by transforming the linear predictors by the inverse of the link function.
  - link – the link object used.
  - linear_predictors – the linear fit on link scale.
  - aic – A version of Akaike's An Information Criterion, minus twice the maximized log-likelihood plus twice the number of parameters.
  - weights – vector of estimated weights for non-probability sample.
  - prior.weights – the weights initially supplied, a vector of 1s if none were.
  - est_totals – the estimated total values of auxiliary variables derived from a non-probability sample.
  - formula – the formula supplied.
  - df_residual – the residual degrees of freedom.
  - log_likelihood – value of log-likelihood function if mle method, in the other case NA.
  - cve – the error for each value of the lambda, averaged across the cross-validation folds for the variable selection model when the propensity score model is fitting. Returned only if selection of variables for the model is used.

  - `method_selection` – Link function, e.g. `logit`, `cloglog` or `probit`.
  - `hessian` – Hessian Gradient of the log-likelihood function from `mle` method.
  - `gradient` – Gradient of the log-likelihood function from `mle` method.
  - `method` – An estimation method for selection model, e.g. `mle` or `gee`.
  - `prob_der` – Derivative of the inclusion probability function for units in a non–probability sample.
  - `prob_rand` – Inclusion probabilities for unit from a probabiliy sample from `svydesign` object.
  - `prob_rand_est` – Inclusion probabilites to a non–probabiliy sample for unit from probability sample.
  - `prob_rand_est_der` – Derivative of the inclusion probabilites to a non–probabiliy sample for unit from probability sample.
- `stat` – matrix of the estimated population means in each bootstrap iteration. Returned only if a bootstrap method is used to estimate the variance and `keep_boot` in `controlInf()` is set on `TRUE`.

Author(s)

Łukasz Chrostowski, Maciej Beręsewicz

References

Kim JK, Park S, Chen Y, Wu C. Combining non-probability and probability survey samples through mass imputation. J R Stat Soc Series A. 2021;184:941– 963.

Shu Yang, Jae Kwang Kim, Rui Song. Doubly robust inference when combining probability and non-probability samples with high dimensional data. J. R. Statist. Soc. B (2020)

Yilin Chen , Pengfei Li & Changbao Wu (2020) Doubly Robust Inference With Nonprobability Survey Samples, Journal of the American Statistical Association, 115:532, 2011-2021

Shu Yang, Jae Kwang Kim and Youngdeok Hwang Integration of data from probability surveys and big found data for finite population inference using mass imputation. Survey Methodology, June 2021 29 Vol. 47, No. 1, pp. 29-58

See Also

`stats::optim()` – For more information on the `optim` function used in the `optim` method of propensity score model fitting.

`maxLik::maxLik()` – For more information on the `maxLik` function used in `maxLik` method of propensity score model fitting.

`ncvreg::cv.ncvreg()` – For more information on the `cv.ncvreg` function used in variable selection for the outcome model.

`nleqslv::nleqslv()` – For more information on the `nleqslv` function used in estimation process of the bias minimization approach.

`stats::glm()` – For more information about the generalised linear models used during mass imputation process.

`RANN::nn2()` – For more information about the nearest neighbour algorithm used during mass imputation process.

controlSel() – For the control parameters related to selection model.

controlOut() – For the control parameters related to outcome model.

controlInf() – For the control parameters related to statistical inference.

Examples

```
# generate data based on Doubly Robust Inference With Non-probability Survey Samples (2021)
# Yilin Chen , Pengfei Li & Changbao Wu
library(sampling)
set.seed(123)
# sizes of population and probability sample
N <- 20000 # population
n_b <- 1000 # probability
# data
z1 <- rbinom(N, 1, 0.7)
z2 <- runif(N, 0, 2)
z3 <- rexp(N, 1)
z4 <- rchisq(N, 4)

# covariates
x1 <- z1
x2 <- z2 + 0.3 * z2
x3 <- z3 + 0.2 * (z1 + z2)
x4 <- z4 + 0.1 * (z1 + z2 + z3)
epsilon <- rnorm(N)
sigma_30 <- 10.4
sigma_50 <- 5.2
sigma_80 <- 2.4

# response variables
y30 <- 2 + x1 + x2 + x3 + x4 + sigma_30 * epsilon
y50 <- 2 + x1 + x2 + x3 + x4 + sigma_50 * epsilon
y80 <- 2 + x1 + x2 + x3 + x4 + sigma_80 * epsilon

# population
sim_data <- data.frame(y30, y50, y80, x1, x2, x3, x4)
## propensity score model for non-probability sample (sum to 1000)
eta <- -4.461 + 0.1 * x1 + 0.2 * x2 + 0.1 * x3 + 0.2 * x4
rho <- plogis(eta)

# inclusion probabilities for probability sample
z_prob <- x3 + 0.2051
sim_data$p_prob <- inclusionprobabilities(z_prob, n = n_b)

# data
sim_data$flag_nonprob <- UPpoisson(rho) ## sampling nonprob
sim_data$flag_prob <- UPpoisson(sim_data$p_prob) ## sampling prob
nonprob_df <- subset(sim_data, flag_nonprob == 1) ## non-probability sample
svyprob <- svydesign(
  ids = ~1, probs = ~p_prob,
  data = subset(sim_data, flag_prob == 1),
  pps = "brewer"
```

```
) ## probability sample

## mass imputation estimator
MI_res <- nonprob(
  outcome = y80 ~ x1 + x2 + x3 + x4,
  data = nonprob_df,
  svydesign = svyprob
)
summary(MI_res)
## inverse probability weighted estimator
IPW_res <- nonprob(
  selection = ~ x1 + x2 + x3 + x4,
  target = ~y80,
  data = nonprob_df,
  svydesign = svyprob
)
summary(IPW_res)
## doubly robust estimator
DR_res <- nonprob(
  outcome = y80 ~ x1 + x2 + x3 + x4,
  selection = ~ x1 + x2 + x3 + x4,
  data = nonprob_df,
  svydesign = svyprob
)
summary(DR_res)
```

`pop.size` *Estimate size of population*

Description

Estimate size of population

Usage

```
pop.size(object, ...)
```

Arguments

`object`	object returned by `nonprobsvy`.
`...`	additional parameters

Value

Vector returning the value of the estimated population size.

probit_model_nonprobsvy *Probit model for weights adjustment*

Description

probit_model_nonprobsvy returns all the methods/objects/functions required to estimate the model, assuming a probit link function.

Usage

```
probit_model_nonprobsvy(...)
```

Arguments

...	Additional, optional arguments.

Value

List with selected methods/objects/functions.

Author(s)

Łukasz Chrostowski, Maciej Beręsewicz

See Also

nonprob() – for fitting procedure with non-probability samples.

summary.nonprobsvy *Summary statistics for model of nonprobsvy class.*

Description

Summary statistics for model of nonprobsvy class.

Usage

```
## S3 method for class 'nonprobsvy'
summary(object, test = c("t", "z"), correlation = FALSE, cov = NULL, ...)
```

Arguments

object	object of nonprobsvy class
test	Type of test for significance of parameters "t" for t-test and "z" for normal approximation of students t distribution, by default "z" is used if there are more than 30 degrees of freedom and "t" is used in other cases.
correlation	correlation Logical value indicating whether correlation matrix should be computed from covariance matrix by default FALSE.
cov	Covariance matrix corresponding to regression parameters
...	Additional optional arguments

Value

An object of summary_nonprobsvy class containing:

- call – A call which created object.
- pop_total – A list containing information about the estimated population mean, its standard error and confidence interval.
- sample_size – The size of the samples used in the model.
- population_size – The estimated size of the population from which the non–probability sample was drawn.
- test – Type of statistical test performed.
- control – A List of control parameters used in fitting the model.
- model – A descriptive name of the model used, e.g., "Doubly-Robust", "Inverse probability weighted", or "Mass Imputation".
- aic – Akaike's information criterion.
- bic – Bayesian (Schwarz's) information criterion.
- residuals – Residuals from the model, providing information on the difference between observed and predicted values.
- likelihood – Logarithm of likelihood function evaluated at coefficients.
- df_residual – Residual degrees of freedom.
- weights – Distribution of estimated weights obtained from the model.
- coef – Regression coefficients estimated by the model.
- std_err – Standard errors of the regression coefficients.
- w_val – Wald statistic values for the significance testing of coefficients.
- p_values – P-values corresponding to the Wald statistic values, assessing the significance of coefficients.
- crr – The correlation matrix of the model coefficients, if requested.
- confidence_interval_coef – Confidence intervals for the model coefficients.
- names – Names of the fitted models.

vcov.nonprobsvy *Obtain Covariance Matrix estimation.*

Description

A vcov method for nonprobsvy class.

Usage

```
## S3 method for class 'nonprobsvy'
vcov(object, ...)
```

Arguments

object	object of nonprobsvy class.
...	additional arguments for method functions

Details

Returns a estimated covariance matrix for model coefficients calculated from analytic hessian or Fisher information matrix usually utilising asymptotic effectiveness of maximum likelihood estimates.

Value

A covariance matrix for fitted coefficients

Index

cloglog_model_nonprobsvy, 2
confint.nonprobsvy, 3
controlInf, 3
controlInf(), *14–17*
controlOut, 4
controlOut(), *14*, *17*
controlSel, 6
controlSel(), *14*, *17*

genSimData, 8

logit_model_nonprobsvy, 9

maxLik::maxLik(), *7*, *16*

ncvreg::cv.ncvreg(), *16*
nleqslv::nleqslv(), *7*, *16*
nonprob, 10
nonprob(), *2*, *4*, *6*, *8, 9*, *19*

pop.size, 18
probit_model_nonprobsvy, 19

RANN::nn2(), *5*, *13*, *15, 16*

stats::glm(), *13*, *15, 16*
stats::optim(), *7*, *16*
summary.nonprobsvy, 19
survey::as.svrepdesign(), *4*

vcov.nonprobsvy, 21