Package: nonprobsvy (via r-universe)

October 23, 2026

Type Package

Title Inference Based on Non-Probability Samples

Version 0.3.0

Description Statistical inference with non-probability samples when auxiliary information from external sources such as probability samples or population totals or means is available. The package implements various methods such as inverse probability (propensity score) weighting, mass imputation and doubly robust approach. Details can be found in: Chen et al. (2020) <doi:10.1080/01621459.2019.1677241>, Yang et al. (2020) <doi:10.1111/rssb.12354>, Kim et al. (2021) <doi:10.1111/rssa.12696>, Yang et al. (2021) <https://www150.statcan.gc.ca/n1/pub/12-001-x/2021001/article/00004-eng.htm> and Wu (2022) <https://www150.statcan.gc.ca/n1/pub/12-001-x/2022002/article/00002-eng.htm>. For details on the package and its functionalities see <doi:10.18637/jss.v117.i02>.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

URL https://github.com/ncn-foreigners/nonprobsvy, https://ncn-foreigners.ue.poznan.pl/nonprobsvy/

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

Depends R (>= 4.0.0), survey

Imports doRNG, maxLik, stats, Matrix, MASS, ncvreg, RANN, Rcpp (>= 1.0.12), nleqslv, doParallel, foreach, parallel, formula.tools

Suggests tinytest, covr, spelling, ggplot2, knitr, rmarkdown

LinkingTo Rcpp, RcppArmadillo

VignetteBuilder knitr

Language en-US

NeedsCompilation yes

Author Łukasz Chrostowski [aut, ctb], Maciej Beręsewicz [aut, cre] (ORCID: <https://orcid.org/0000-0002-8281-4301>), Piotr Chlebicki [aut, ctb] (ORCID: <https://orcid.org/0009-0006-4867-7434>)

Maintainer Maciej Beręsewicz <maciej.beresewicz@ue.poznan.pl>

Repository https://cran.r-universe.dev

Date/Publication 2026-09-23 17:16:15 UTC

RemoteUrl https://github.com/cran/nonprobsvy

RemoteRef HEAD

RemoteSha 655a68e3eeb8cb9b0a885e8bfa54be52639f8735

Contents

admin . . . 3
check_balance . . . 3
coef.nonprob . . . 4
confint.nonprob . . . 5
control_inf . . . 6
control_out . . . 8
control_sel . . . 10
extract . . . 12
jvs . . . 12
method_glm . . . 13
method_nn . . . 16
method_npar . . . 19
method_pmm . . . 21
method_ps . . . 25
nobs.nonprob . . . 26
nonprob . . . 27
plot.nonprob . . . 36
pop_size . . . 37
print.nonprob_summary . . . 38
summary.nonprob . . . 38
update.nonprob . . . 40
weights.nonprob . . . 41

Index **43**

admin *Admin Data (Non-Probability Survey)*

Description

This is a subset of the Central Job Offers Database, a voluntary administrative data set (non-probability sample). The data was slightly manipulated to ensure the relationships were preserved, and then aligned. For more information about the CBOP, please refer to: https://oferty.praca.gov.pl/portal.

Usage

```
admin
```

Format

A single data.frame with 9,344 rows and 6 columns

`id` Identifier of an entity (company: legal or local).

`private` Whether the company is a private (1) or public (0) entity.

`size` The size of the entity: S – small (to 9 employees), M – medium (10-49) or L – large (over 49).

`nace` The main NACE code for a given entity: C, D.E, F, G, H, I, J, K.L, M, N, O, P, Q or R.S (14 levels, 3 combined: D and E, K and L, and R and S).

`region` The region of Poland (16 levels: 02, 04, ..., 32).

`single_shift` Whether an entity seeks employees on a single shift.

Examples

```
data("admin")
head(admin)
```

---

check_balance *Checks Variable Balance Between the Probability and Non-Probability Samples*

Description

Function compares totals for auxiliary variables specified in the `x` argument for an `object` that contains either IPW or DR estimator.

Usage

```
check_balance(x, object, dig)
```

Arguments

x	formula specifying variables to check
object	object of nonprob class
dig	number of digits for rounding (default = 2)

Value

A list containing totals for non-probability and probability samples and their differences

Examples

```
sample_a <- data.frame(y = c(1, 0, 1, 0, 1), x = c(0, 1, 2, 3, 4))
sample_b <- data.frame(x = c(0.5, 1.5, 2.5, 3.5), w = c(4, 4, 4, 4))
sample_b_svy <- svydesign(ids = ~1, weights = ~w, data = sample_b)

ipw_est1 <- nonprob(
  selection = ~x,
  target = ~y,
  svydesign = sample_b_svy,
  data = sample_a,
  method_selection = "logit",
  se = FALSE
)

ipw_est2 <- nonprob(
  selection = ~x,
  target = ~y,
  svydesign = sample_b_svy,
  data = sample_a,
  method_selection = "logit",
  control_selection = control_sel(est_method = "gee", gee_h_fun = 1),
  se = FALSE
)

## check the balance for the standard IPW
check_balance(~x, ipw_est1)

## check the balance for the calibrated IPW
check_balance(~x, ipw_est2)
```

---

coef.nonprob *Returns Coefficients of the Underlying Models*

Description

Returns a list of coefficients for the selection and the outcome models

Usage

```
## S3 method for class 'nonprob'
coef(object, ...)
```

Arguments

object	a nonprob class object
...	other arguments passed to methods (currently not supported)

Value

a list with two entries:

- "coef_sel" a matrix of coefficients for the selection equation if possible, else NULL
- "coef_dr" a matrix of coefficients for the outcome equation(s) if possible, else NULL

Examples

```
data(admin)
data(jvs)

jvs_svy <- svydesign(ids = ~ 1,  weights = ~ weight,
strata = ~ size + nace + region, data = jvs)

ipw_est1 <- nonprob(selection = ~ region + private + nace + size,
target = ~ single_shift,
svydesign = jvs_svy,
data = admin, method_selection = "logit", se = FALSE
)

coef(ipw_est1)
```

---

confint.nonprob *Returns Confidence Intervals for Estimated Mean*

Description

A generic function that returns the confidence interval for the estimated mean. If standard errors have not been estimated, the function updates the nonprob object to obtain standard errors.

Usage

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

Arguments

object	object of nonprob 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

returns a data.frame with confidence intervals for the target variables

Examples

```
data(admin)
data(jvs)

jvs_svy <- svydesign(ids = ~ 1,  weights = ~ weight,
strata = ~ size + nace + region, data = jvs)

ipw_est1 <- nonprob(selection = ~ region + private + nace + size,
target = ~ single_shift,
svydesign = jvs_svy,
data = admin, method_selection = "logit", se = FALSE
)

confint(ipw_est1)
```

---

control_inf *Control Parameters for Inference*

Description

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

Usage

```
control_inf(
  var_method = c("analytic", "bootstrap"),
 rep_type = c("subbootstrap", "auto", "JK1", "JKn", "BRR", "bootstrap", "mrbbootstrap",
    "Fay"),
  vars_selection = FALSE,
  vars_combine = FALSE,
  bias_correction = FALSE,
  num_boot = 100,
  alpha = 0.05,
  cores = 1,
```

```
  keep_boot = TRUE,
  nn_exact_se = FALSE
)
```

Arguments

var_method	the variance method (default "analytic"). Note that the doubly robust analytic variance is derived under the logistic propensity model (Chen, Li & Wu 2020, Theorem 2); for the "probit" and "cloglog" selection links it is a conservative approximation (it tends to over-estimate the standard error and can be numerically unstable when fitted propensities approach 1), so var_method = "bootstrap" is recommended for doubly robust inference with those links.
rep_type	the replication type for weights in the bootstrap method for variance estimation passed to survey::as.svrepdesign(). Default is "subbootstrap".
vars_selection	logical scalar (default FALSE); if TRUE, then the variables selection model is used.
vars_combine	logical scalar indicating whether variables should be combined after variable selection for doubly robust estimators (default FALSE)
bias_correction	logical scalar (default FALSE); if TRUE, the doubly-robust mean is estimated by jointly solving the Yang-Kim-Song (2020) eq. (9) system in (theta, beta). When vars_selection = FALSE this gives the low-dimensional Kim & Haziza (2014) joint estimator; when vars_selection = TRUE and vars_combine = TRUE it gives the high-dimensional two-step Yang-Kim-Song estimator. A svydesign argument is required (joint estimation needs individual-level probability sample data).
num_boot	the number of iteration for bootstrap algorithms.
alpha	significance level (default 0.05).
cores	the number of cores in parallel computing (default 1).
keep_boot	a logical value indicating whether statistics from bootstrap should be kept (default TRUE)
nn_exact_se	a 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 computed using covariance between imputed values for units in the probability sample using predictive matches from the non-probability sample. In most situations this term is negligible and is very computationally expensive so by default it is set to FALSE, but the recommended option is to set this value to TRUE before submitting the final results.

Value

A list with selected parameters.

See Also

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

control_out *Control Parameters for Outcome Model*

Description

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

Usage

```
control_out(
  epsilon = 1e-08,
  maxit = 100,
  trace = FALSE,
  k = 5,
  penalty = c("SCAD", "lasso", "MCP"),
  a_SCAD = 3.7,
  a_MCP = 3,
  lambda = -1,
  lambda_min = 0.001,
  nlambda = 50,
  nfolds = 10,
  treetype = c("kd", "bd"),
  searchtype = c("standard", "priority"),
  pmm_match_type = 1,
  pmm_weights = c("none", "dist"),
  pmm_k_choice = c("none", "min_var"),
  pmm_k_max = NULL,
  pmm_reg_engine = c("glm", "loess"),
 npar_loess = stats::loess.control(surface = "direct", trace.hat = "approximate")
)
```

Arguments

epsilon	Tolerance for fitting algorithms. Default is 1e-8.
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	A user-specified λ value during variable selection model fitting. The default value -1 uses cross-validation.
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 during cross-validation for variables selection model.

`treetype` Type of tree for nearest neighbour imputation (for the NN and PMM estimator) passed to `RANN::nn2()` function.

`searchtype` Type of search for nearest neighbour imputation (for the NN and PMM estimator) passed to `RANN::nn2()` function.

`pmm_match_type` (Only for the PMM Estimator) Indicates how to select 'closest' unit from non-probability sample for each unit in probability sample. Either `1` (default) or `2` where `2` is matching by minimizing distance between $\hat{y}_i$ for $i \in S_{\mathrm{NP}}$ and y_j for $j \in S_{\mathrm{P}}$ and `1` is matching by minimizing distance between $\hat{y}_i$ for $i \in S_{\mathrm{NP}}$ and $\hat{y}_i$ for $i \in S_{\mathrm{NP}}$.

`pmm_weights` (Only for the PMM Estimator) Indicate how to weight `k` nearest neighbours in S_{P} to create imputed value for units in S_{NP}. The default value `"none"` indicates that mean of `k` nearest y's from S_{P} should be used whereas `"dist"` results in weighted mean of these `k` values where weights are inversely proportional to distance between matched values.

`pmm_k_choice` (Only for the PMM Estimator) 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 a full search over `1:n_NP` (or `1:pmm_k_max`, see below), where n_{NP} is the non-probability sample size, minimizing the estimated variance of the mean estimator. The `k` value supplied in this control list is replaced by the selected value. Note that this search refits the full PMM stack for every candidate `k`, so its cost scales as $O(n_{\mathrm{NP}} \times l)$ (with l the number of outcome variables) and can be substantial for large non-probability samples; cap it with `pmm_k_max` or supply `k` directly when n_{NP} is large.

`pmm_k_max` (Only for the PMM Estimator) Positive integer upper bound for the `pmm_k_choice = "min_var"` search grid. The default `NULL` searches the full `1:n_NP` grid. Setting e.g. `pmm_k_max = 30` caps the search at `1:min(n_NP, 30)` to bound its cost.

`pmm_reg_engine` (Only for the PMM Estimator) whether to use parametric (`"glm"`) or non-parametric (`"loess"`) regression model for the outcome. The default is `"glm"`.

`npar_loess` control parameters for the stats::loess via the stats::loess.control function.

Value

List with selected parameters.

See Also

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

control_sel *Control Parameters for the Selection Model*

Description

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

Usage

```
control_sel(
  est_method = c("mle", "gee"),
  gee_h_fun = 1,
  optimizer = c("maxLik", "optim"),
  maxlik_method = c("NR", "BFGS", "NM"),
  optim_method = c("BFGS", "Nelder-Mead"),
  epsilon = 1e-04,
  maxit = 500,
  trace = FALSE,
  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("zero", "mle", "naive"),
  nleqslv_method = c("Broyden", "Newton"),
 nleqslv_global = c("dbldog", "pwldog", "cline", "qline", "gline", "hook", "none"),
  nleqslv_xscalm = c("fixed", "auto"),
  dependence = FALSE,
  key = NULL
)
```

Arguments

est_method	Method of estimation for propensity score model ("mle" or "gee"; default is "mle").
gee_h_fun	Smooth function for the generalized estimating equations (GEE) method.
optimizer	(for the est_method="mle" only) optimization function for maximum likelihood estimation.
maxlik_method	(for the est_method="mle" only) maximisation method that will be passed to maxLik::maxLik() function. Default is NR.
optim_method	(for the est_method="mle" only) maximisation method that will be passed to stats::optim() function. Default is BFGS.

epsilon	Tolerance for fitting algorithms by default 1e-4.
maxit	Maximum number of iterations.
trace	logical value. If TRUE trace steps of the fitting algorithms. Default is FALSE
penalty	The penalization 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 zero then start is a vector of zeros (default for all methods). – if mle (for est_method="gee" only) starting parameters are taken from the result of the est_method="mle" method.
nleqslv_method	(for the est_method="gee" only) The method that will be passed to nleqslv::nleqslv() function.
nleqslv_global	(for the est_method="gee" only) The global strategy that will be passed to nleqslv::nleqslv() function.
nleqslv_xscalm	(for the est_method="gee" only) The type of x scaling that will be passed to nleqslv::nleqslv() function.
dependence	logical value (default TRUE) informing whether samples overlap (NOT YET IMPLEMENTED, FOR FUTURE DEVELOPMENT).
key	binary key variable allowing to identify the overlap (NOT YET IMPLEMENTED, FOR FUTURE DEVELOPMENT).

Details

Smooth function (gee_h_fun) for the generalized estimating equations (GEE) method taking the following values

- if 1 then $\boldsymbol{h}(\boldsymbol{x}, \boldsymbol{\theta}) = \frac{\boldsymbol{x}}{\pi(\boldsymbol{x}, \boldsymbol{\theta})}$,
- if 2 then $\boldsymbol{h}(\boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{x}$

Value

List with selected parameters.

See Also

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

extract *Extracts Estimates from the Nonprob Class Object*

Description

Returns a data.frame of estimated mean(s) or standard error(s)

Usage

```
extract(object, what)
```

Arguments

object	object of of the nonprob class
what	what to extract: all estimates (mean(s), SE(s) and CI(s); "all"; default), estimated mean(s) ("mean") or their standard error(s) ("se")

Value

a data.frame with selected information

Examples

```
data(admin)
data(jvs)

jvs_svy <- svydesign(ids = ~ 1,  weights = ~ weight,
strata = ~ size + nace + region, data = jvs)

ipw_est1 <- nonprob(selection = ~ region + private + nace + size,
target = ~ single_shift,
svydesign = jvs_svy,
data = admin, method_selection = "logit"
)
extract(ipw_est1)
extract(ipw_est1, "se")
```

jvs *Job Vacancy Survey*

Description

This is a subset of the Job Vacancy Survey from Poland (for one quarter). The data has been subject to slight manipulation, but the relationships in the data have been preserved. For further details on the JVS, please refer to the following link: https://stat.gov.pl/obszary-tematyczne/rynek-pracy/popyt-na-prace/zeszyt-metodologiczny-popyt-na-prace,3,1.html.

Usage

```
jvs
```

Format

A single data.frame with 6,523 rows and 6 columns

`id` Identifier of an entity (company: legal or local).

`private` Whether the company is a private (1) or public (0) entity.

`size` The size of the entity: S – small (to 9 employees), M – medium (10-49) or L – large (over 49).

`nace` The main NACE code for a given entity: C, D.E, F, G, H, I, J, K.L, M, N, O, P, Q or R.S (14 levels, 3 combined: D and E, K and L, and R and S).

`region` The region of Poland (16 levels: 02, 04, ..., 32).

`weight` The final (calibrated) weight (w-weight). We do not have access to design weights (d-weights).

Examples

```
data("jvs")
head(jvs)
```

---

method_glm *Mass Imputation Using the Generalized Linear Model Method*

Description

Model for the outcome for the mass imputation estimator using generalized linear models via the `stats::glm` function. Estimation of the mean is done using S_{P} probability sample or known population totals.

Usage

```
method_glm(
  y_nons,
  X_nons,
  X_rand,
  svydesign,
  weights = NULL,
  family_outcome = "gaussian",
  start_outcome = NULL,
  vars_selection = FALSE,
  pop_totals = NULL,
  pop_size = NULL,
  control_outcome = control_out(),
```

```
  control_inference = control_inf(),
  verbose = FALSE,
  se = TRUE
)
```

Arguments

y_nons	target variable from non-probability sample
X_nons	a model.matrix with auxiliary variables from non-probability sample
X_rand	a model.matrix with auxiliary variables from non-probability sample
svydesign	a svydesign object
weights	case / frequency weights from non-probability sample
family_outcome	family for the glm model
start_outcome	start parameters (default NULL)
vars_selection	whether variable selection should be conducted
pop_totals	population totals from the nonprob function
pop_size	population size from the nonprob function
control_outcome	controls passed by the control_out function
control_inference	controls passed by the control_inf function (currently not used, for further development)
verbose	parameter passed from the main nonprob function
se	whether standard errors should be calculated

Details

Analytical variance

The variance of the mean is estimated based on the following approach

(a) non-probability part (S_{NP} with size n_{NP}; denoted as var_nonprob in the result)

$$\hat{V}_1 = \frac{1}{n_{\mathrm{NP}}^2} \sum_{i=1}^{n_{\mathrm{NP}}} \hat{e}_i^2 \left\{ \boldsymbol{h}(\boldsymbol{x}_i; \hat{\boldsymbol{\beta}})' \hat{\boldsymbol{c}} \right\}^2,$$

where $\hat{e}_i = y_i - m(\boldsymbol{x}_i; \hat{\boldsymbol{\beta}})$ and

$$\widehat{\boldsymbol{c}} = \left\{ n_{\mathrm{NP}}^{-1} \sum_{i \in S_{\mathrm{NP}}} \dot{\boldsymbol{m}} \left(\boldsymbol{x}_i; \boldsymbol{\beta}^* \right) \boldsymbol{h} \left(\boldsymbol{x}_i; \boldsymbol{\beta}^* \right)' \right\}^{-1} N^{-1} \sum_{i \in S_{\mathrm{P}}} w_i \dot{\boldsymbol{m}} \left(\boldsymbol{x}_i; \boldsymbol{\beta}^* \right).$$

Under the linear regression model $\boldsymbol{h}\left(\boldsymbol{x}_i; \widehat{\boldsymbol{\beta}}\right) = \boldsymbol{x}_i$ and $\widehat{\boldsymbol{c}} = \left(n_{\mathrm{NP}}^{-1} \sum_{i \in S_{\mathrm{NP}}} \boldsymbol{x}_i \boldsymbol{x}_i' \right)^{-1} N^{-1} \sum_{i \in S_{\mathrm{P}}} w_i \boldsymbol{x}_i$.

(b) probability part (S_{P} with size n_{P}; denoted as var_prob in the result)

This part uses functionalities of the {survey} package and the variance is estimated using the following equation:

$$\hat{V}_2 = \frac{1}{N^2} \sum_{i=1}^{n_{\mathrm{P}}} \sum_{j=1}^{n_{\mathrm{P}}} \frac{\pi_{ij} - \pi_i \pi_j}{\pi_{ij}} \frac{m(\boldsymbol{x}_i; \hat{\boldsymbol{\beta}})}{\pi_i} \frac{m(\boldsymbol{x}_i; \hat{\boldsymbol{\beta}})}{\pi_j}.$$

Note that $\hat{V}_2$ in principle can be estimated in various ways depending on the type of the design and whether population size is known or not.

Furthermore, if only population totals/means are known and assumed to be fixed we set $\hat{V}_2 = 0$.

Information on the case when svydesign is not available:

1. variance is estimated only for the non-probability part with $\hat{V}_1$ defined above.
2. point estimator of $\hat{\mu}_y$ for linear regression is estimated using $\mu_x' \hat{\boldsymbol{\beta}}$ where μ_x is the vector of population means
3. for non-linear functions such as logistic or Poisson regression we use a simplification, i.e. we report point estimate as $\exp(\mu_x' \hat{\boldsymbol{\beta}})$ for Poisson and $\frac{\exp(\mu_x' \hat{\boldsymbol{\beta}})}{1+\exp(\mu_x' \hat{\boldsymbol{\beta}})}$ for logistic regression.

Value

an nonprob_method class which is a list with the following entries

model_fitted fitted model either a glm.fit, cv.ncvreg, or ncvreg object

y_nons_pred predicted values for the non-probablity sample

y_rand_pred predicted values for the probability sample or population totals

coefficients coefficients for the model (if available)

svydesign an updated surveydesign2 object (new column y_hat_MI is added)

y_mi_hat estimated population mean for the target variable

vars_selection whether variable selection was performed

var_prob variance for the probability sample component (if available)

var_nonprob variance for the non-probability sampl component

var_total total variance, if possible it should be var_prob+var_nonprob if not, just a scalar

model model type (character "glm")

family family type (character "glm")

References

Kim, J. K., Park, S., Chen, Y., & Wu, C. (2021). Combining non-probability and probability survey samples through mass imputation. Journal of the Royal Statistical Society Series A: Statistics in Society, 184(3), 941-963.

Examples

```
data(admin)
data(jvs)
jvs_svy <- svydesign(ids = ~ 1,  weights = ~ weight, strata = ~ size + nace + region, data = jvs)

res_glm <- method_glm(y_nons = admin$single_shift,
                      X_nons = model.matrix(~ region + private + nace + size, admin),
                      X_rand = model.matrix(~ region + private + nace + size, jvs),
                      svydesign = jvs_svy)

res_glm
```

`method_nn` *Mass Imputation Using Nearest Neighbours Matching Method*

Description

Mass imputation using nearest neighbours approach as described in Yang et al. (2021). The implementation is currently based on RANN::nn2 function and thus it uses Euclidean distance for matching units from S_{NP} (non-probability) to S_{P} (probability). Matching ties are randomized before donor values are aggregated, so tied nearest neighbours are not selected only by input row order. Estimation of the mean is done using S_{P} sample: when `pop_size` is supplied this is the known-N Horvitz-Thompson mean, otherwise it reduces to the usual ratio mean with $\hat{N} = \sum_{i \in S_{\mathrm{P}}} d_{\mathrm{P},i}$. The `pop_size` argument is not converted into a finite population correction; if an fpc is needed, it should be supplied in `svydesign`, where it is handled by the `{survey}` variance routines.

Usage

```
method_nn(
  y_nons,
  X_nons,
  X_rand,
  svydesign,
  weights = NULL,
  family_outcome = NULL,
  start_outcome = NULL,
  vars_selection = FALSE,
  pop_totals = NULL,
  pop_size = NULL,
  control_outcome = control_out(),
  control_inference = control_inf(),
  verbose = FALSE,
  se = TRUE,
  nn_matches = NULL
)
```

Arguments

y_nons	target variable from non-probability sample
X_nons	a model.matrix with auxiliary variables from non-probability sample
X_rand	a model.matrix with auxiliary variables from non-probability sample
svydesign	a svydesign object
weights	case / frequency weights from non-probability sample. If nn_exact_se=TRUE, non-constant weights also define mini-bootstrap sampling probabilities proportional to their inverses.
family_outcome	a placeholder (not used in method_nn)
start_outcome	a placeholder (not used in method_nn)
vars_selection	whether variable selection should be conducted
pop_totals	a placeholder (not used in method_nn)
pop_size	population size from the nonprob function. If NULL, the method uses sum(weights(svydesign)). If supplied, it is used as the known-N denominator for the mean and variance scaling, but it does not modify the finite population correction of svydesign.
control_outcome	controls passed by the control_out function
control_inference	controls passed by the control_inf function
verbose	parameter passed from the main nonprob function
se	whether standard errors should be calculated
nn_matches	optional precomputed nearest-neighbour search results for internal reuse across outcomes. If supplied, it should be a list with rand and nons entries from RANN::nn2().

Details

Analytical variance

The variance of the mean is estimated based on the following approach

(a) non-probability part (S_{NP} with size n_{NP}; denoted as var_nonprob in the result)

This may be estimated using

$$\hat{V}_1 = \frac{1}{N^2} \sum_{i=1}^{S_{\mathrm{NP}}} \frac{1 - \hat{\pi}_{\mathrm{P}}(\boldsymbol{x}_i)}{\hat{\pi}_{\mathrm{P}}(\boldsymbol{x}_i)} \hat{\sigma}^2(\boldsymbol{x}_i),$$

where $\hat{\pi}_{\mathrm{P}}(\boldsymbol{x}_i)$ is an estimator of propensity scores which we currently estimate using n_{NP}/N (constant) and $\hat{\sigma}^2(\boldsymbol{x}_i)$ is estimated using based on the average of $(y_i - y_i^*)^2$. The y_i^* values used in this proxy are obtained by leave-one-out matching in S_{NP}, so a unit is not used as its own donor.

Chlebicki et al. (2025, Algorithm 2) proposed non-parametric mini-bootstrap estimator (without assuming that it is consistent) but with good finite population properties. This bootstrap can be applied using control_inference(nn_exact_se=TRUE) and can be summarized as follows:

1. Sample n_{NP} units from S_{NP} with replacement to create S'_{NP}. If non-constant pseudo-weights are supplied through `weights`, sampling probabilities are proportional to their inverses; equal weights use uniform resampling.
2. Match units from S_{P} to S'_{NP} to obtain predictions y^*=$k^{-1}\sum_k y_k$.
3. Estimate $\hat{\mu} = \frac{1}{N}\sum_{i\in S_{\text{P}}} d_{\text{P},i} y_i^*$.
4. Repeat steps 1-3 M times (we set $M = 50$ in our simulations; this is hard-coded).
5. Estimate $\hat{V}_1 = \text{var}(\hat{\boldsymbol{\mu}})$ obtained from simulations and save it as `var_nonprob`.

(b) probability part (S_{P} with size n_{P}; denoted as `var_prob` in the result)

This part uses functionalities of the `{survey}` package and the variance is estimated using the following equation:

$$\hat{V}_2 = \frac{1}{N^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\frac{\pi_{ij}-\pi_i\pi_j}{\pi_{ij}}\frac{y_i^*}{\pi_i}\frac{y_j^*}{\pi_j},$$

where y_i^* and y_j^* are values imputed imputed as an average of k-nearest neighbour, i.e. $k^{-1}\sum_k y_k$. Note that $\hat{V}_2$ in principle can be estimated in various ways depending on the type of the design and whether population size is known or not.

Value

an `nonprob_method` class which is a `list` with the following entries

model_fitted `RANN::nn2` object

y_nons_pred predicted values for the non-probablity sample (query to itself)

y_rand_pred predicted values for the probability sample

coefficients coefficients for the model (if available)

svydesign an updated `surveydesign2` object (new column `y_hat_MI` is added)

y_mi_hat estimated population mean for the target variable

vars_selection whether variable selection was performed (not implemented, for further development)

var_prob variance for the probability sample component (if available)

var_nonprob variance for the non-probability sample component

var_total total variance, if possible it should be `var_prob+var_nonprob` if not, just a scalar

model model type (character `"nn"`)

family placeholder for the `NN approach` information

References

Yang, S., Kim, J. K., & Hwang, Y. (2021). 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

Chlebicki, P., Chrostowski, Ł., & Beręsewicz, M. (2025). Data integration of non-probability and probability samples with predictive mean matching. arXiv preprint arXiv:2403.13750.

Examples

```
sample_a <- data.frame(y = c(1, 0, 1, 0, 1), x = c(0, 1, 2, 3, 4))
sample_b <- data.frame(x = c(0.5, 1.5, 2.5, 3.5), w = c(4, 4, 4, 4))
sample_b_svy <- svydesign(ids = ~1, weights = ~w, data = sample_b)

res_nn <- method_nn(
  y_nons = sample_a$y,
  X_nons = model.matrix(~x, sample_a),
  X_rand = model.matrix(~x, sample_b),
  svydesign = sample_b_svy,
  control_outcome = control_out(k = 1),
  se = FALSE
)

res_nn
```

`method_npar` *Mass Imputation Using Non-Parametric Model Method*

Description

Model for the outcome for the mass imputation estimator using loess via `stats::loess`. Estimation of the mean is done using the S_{P} probability sample.

Usage

```
method_npar(
  y_nons,
  X_nons,
  X_rand,
  svydesign,
  weights = NULL,
  family_outcome = "gaussian",
  start_outcome = NULL,
  vars_selection = FALSE,
  pop_totals = NULL,
  pop_size = NULL,
  control_outcome = control_out(),
  control_inference = control_inf(),
  verbose = FALSE,
  se = TRUE
)
```

Arguments

`y_nons`	target variable from non-probability sample
`X_nons`	a `model.matrix` with auxiliary variables from non-probability sample

X_rand	a model.matrix with auxiliary variables from non-probability sample
svydesign	a svydesign object
weights	case / frequency weights from non-probability sample (default NULL)
family_outcome	family for the glm model
start_outcome	a place holder (not used in method_npar)
vars_selection	whether variable selection should be conducted
pop_totals	a place holder (not used in method_npar)
pop_size	population size from the nonprob function
control_outcome	controls passed by the control_out function
control_inference	controls passed by the control_inf function
verbose	parameter passed from the main nonprob function
se	whether standard errors should be calculated

Details

Analytical variance

The variance of the mean is estimated based on the following approach

(a) non-probability part (S_{NP} with size n_{NP}; denoted as var_nonprob in the result)

$$\hat{V}_1 = \frac{1}{N^2} \sum_{i=1}^{n_{\mathrm{NP}}} \left\{\hat{g}_{\mathrm{P}}(\boldsymbol{x}_i)\right\}^2 \hat{e}_i^2,$$

where $\hat{e}_i = y_i - \hat{m}(x_i)$ is the residual and $\hat{g}_{\mathrm{P}}(\boldsymbol{x}_i) = \left\{\pi_{\mathrm{P}}(\boldsymbol{x}_i)\right\}^{-1}$ can be estimated various ways. In the package we estimate $\hat{g}_{\mathrm{P}}(\boldsymbol{x}_i)$ using $\pi_{\mathrm{P}}(\boldsymbol{x}_i) = E(I_{\mathrm{NP}}|\boldsymbol{x})$ as suggested by Chen et al. (2022, p. 6). In particular, we currently support this using stats::loesswith"gaussian"‘ family.

(b) probability part (S_{P} with size n_{P}; denoted as var_prob in the result)

This part uses functionalities of the {survey} package and the variance is estimated using the following equation:

$$\hat{V}_2 = \frac{1}{N^2} \sum_{i=1}^{n_{\mathrm{P}}} \sum_{j=1}^{n_{\mathrm{P}}} \frac{\pi_{ij} - \pi_i \pi_j}{\pi_{ij}} \frac{\hat{m}(x_i)}{\pi_i} \frac{\hat{m}(x_j)}{\pi_j}.$$

Note that $\hat{V}_2$ in principle can be estimated in various ways depending on the type of the design and whether population size is known or not.

Value

an nonprob_method class which is a list with the following entries

model_fitted fitted model object returned by stats::loess

y_nons_pred predicted values for the non-probablity sample

y_rand_pred predicted values for the probability sample or population totals

coefficients coefficients for the model (if available)

svydesign an updated `surveydesign2` object (new column `y_hat_MI` is added)

y_mi_hat estimated population mean for the target variable

vars_selection whether variable selection was performed

var_prob variance for the probability sample component (if available)

var_nonprob variance for the non-probability sampl component

model model type (character `"npar"`)

References

Chen, S., Yang, S., & Kim, J. K. (2022). Nonparametric mass imputation for data integration. Journal of Survey Statistics and Methodology, 10(1), 1-24.

Examples

```
set.seed(123123123)
N <- 10000
n_a <- 500
n_b <- 1000
n_b1 <- 0.7*n_b
n_b2 <- 0.3*n_b
x1 <- rnorm(N, 2, 1)
x2 <- rnorm(N, 2, 1)
y1 <- rnorm(N, 0.3 + 2*x1+ 2*x2, 1)
y2 <- rnorm(N, 0.3 + 0.5*x1^2+ 0.5*x2^2, 1)
strata <- x1 <= 2
pop <- data.frame(x1, x2, y1, y2, strata)
sample_a <- pop[sample(1:N, n_a),]
sample_a$w_a <- N/n_a
sample_a_svy <- svydesign(ids=~1, weights=~w_a, data=sample_a)
pop1 <- subset(pop, strata == TRUE)
pop2 <- subset(pop, strata == FALSE)
sample_b <- rbind(pop1[sample(1:nrow(pop1), n_b1), ],
                  pop2[sample(1:nrow(pop2), n_b2), ])
res_y_npar <- nonprob(outcome = y1 + y2 ~ x1 + x2,
                      data = sample_b,
                      svydesign = sample_a_svy,
                      method_outcome = "npar")
res_y_npar
```

`method_pmm` *Mass Imputation Using Predictive Mean Matching Method*

Description

Model for the outcome for the mass imputation estimator. The implementation is currently based on RANN::nn2 function and thus it uses Euclidean distance for matching units from S_{NP} (non-probability) to S_{P} (probability) based on predicted values from model $\boldsymbol{x}_i$ based either on `method_glm` or `method_npar`. Estimation of the mean is done using S_{P} sample: when `pop_size` is supplied this is the known-N Horvitz-Thompson mean, otherwise it reduces to the usual ratio mean with $\hat{N} = \sum_{i \in S_{\text{P}}} d_{\text{P},i}$. The `pop_size` argument is not converted into a finite population correction; if an fpc is needed, it should be supplied in `svydesign`, where it is handled by the `{survey}` variance routines. Matching ties are randomized by the nearest-neighbour step before donor values are aggregated.

This implementation extends Yang et al. (2021) approach as described in Chlebicki et al. (2025), namely:

pmm_weights if k>1 weighted aggregation of the mean for a given unit is used. We use distance matrix returned by RANN::nn2 function (`pmm_weights` from the `control_out()` function)

nn_exact_se if the non-probability sample is small we recommend using a mini-bootstrap approach to estimate variance from the non-probability sample (`nn_exact_se` from the `control_inf()` function). If non-constant pseudo-weights are supplied, bootstrap samples are drawn with probabilities proportional to inverse weights and the resampled weights are used in each refitted outcome model.

pmm_k_choice the main `nonprob` function allows for dynamic selection of k neighbours based on a full-grid variance minimization procedure over `1:n_NP` (`pmm_k_choice` from the `control_out()` function)

Usage

```
method_pmm(
  y_nons,
  X_nons,
  X_rand,
  svydesign,
  weights = NULL,
  family_outcome = "gaussian",
  start_outcome = NULL,
  vars_selection = FALSE,
  pop_totals = NULL,
  pop_size = NULL,
  control_outcome = control_out(),
  control_inference = control_inf(),
  verbose = FALSE,
  se = TRUE
)
```

Arguments

`y_nons`	target variable from non-probability sample
`X_nons`	a `model.matrix` with auxiliary variables from non-probability sample

X_rand	a model.matrix with auxiliary variables from non-probability sample
svydesign	a svydesign object
weights	case / frequency weights from non-probability sample. If nn_exact_se=TRUE, non-constant weights also define mini-bootstrap sampling probabilities proportional to their inverses and are resampled for each bootstrap refit.
family_outcome	family for the glm model
start_outcome	start parameters
vars_selection	whether variable selection should be conducted
pop_totals	a place holder (not used in method_pmm)
pop_size	population size from the nonprob function. If NULL, the method uses sum(weights(svydesign)). If supplied, it is used as the known-N denominator for the mean and variance scaling, but it does not modify the finite population correction of svydesign.
control_outcome	controls passed by the control_out function
control_inference	controls passed by the control_inf function
verbose	parameter passed from the main nonprob function
se	whether standard errors should be calculated

Details

Matching

In the package we support two types of matching:

1. $\hat{y} - \hat{y}$ matching (default; control_out(pmm_match_type = 1)).
2. $\hat{y} - y$ matching (control_out(pmm_match_type = 2)).

Analytical variance

The variance of the mean is estimated based on the following approach (a) non-probability part (S_{NP} with size n_{NP}; denoted as var_nonprob in the result) is currently estimated using the non-parametric mini-bootstrap estimator proposed by Chlebicki et al. (2025, Algorithm 2). It is not proved to be consistent but with good finite population properties. This bootstrap can be applied using control_inference(nn_exact_se=TRUE) and can be summarized as follows:

1. Sample n_{NP} units from S_{NP} with replacement to create S'_{NP}. If non-constant pseudo-weights are supplied through weights, sampling probabilities are proportional to their inverses; equal weights use uniform resampling.
2. Estimate regression model $\mathbb{E}[Y|\boldsymbol{X}] = m(\boldsymbol{X}, \cdot)$ based on S'_{NP} from step 1, using the resampled weights when supplied.
3. Compute $\hat{\nu}'(i,t)$ for $t = 1, \ldots, k, i \in S_{\text{P}}$ using estimated $m(\boldsymbol{x}', \cdot)$ and $\{(y_j, \boldsymbol{x}_j)|j \in S'_{\text{NP}}\}$.
4. Compute $\frac{1}{k}\sum_{t=1}^{k} y_{\hat{\nu}'(i)}$ using Y values from S'_{NP}.
5. Repeat steps 1-4 M times (we set (hard-coded) $M = 50$ in our code).

6. Estimate $\hat{V}_1 = \text{var}(\hat{\boldsymbol{\mu}})$ obtained from simulations and save it as `var_nonprob`.

(b) probability part (S_{P} with size n_{P}; denoted as `var_prob` in the result)

This part uses functionalities of the `{survey}` package and the variance is estimated using the following equation:

$$\hat{V}_2 = \frac{1}{N^2} \sum_{i=1}^{n_{\text{P}}} \sum_{j=1}^{n_{\text{P}}} \frac{\pi_{ij} - \pi_i \pi_j}{\pi_{ij}} \frac{m(\boldsymbol{x}_i; \hat{\boldsymbol{\beta}})}{\pi_i} \frac{m(\boldsymbol{x}_i; \hat{\boldsymbol{\beta}})}{\pi_j}.$$

Note that $\hat{V}_2$ in principle can be estimated in various ways depending on the type of the design and whether population size is known or not.

Value

an `nonprob_method` class which is a `list` with the following entries

model_fitted fitted model either an `glm.fit` or `cv.ncvreg` object

y_nons_pred predicted values for the non-probablity sample

y_rand_pred predicted values for the probability sample or population totals

coefficients coefficients for the model (if available)

svydesign an updated `surveydesign2` object (new column `y_hat_MI` is added)

y_mi_hat estimated population mean for the target variable

vars_selection whether variable selection was performed

var_prob variance for the probability sample component (if available)

var_nonprob variance for the non-probability sampl component

model model type (character "pmm")

family depends on the method selected for estimating E(Y|X)

Examples

```
sample_a <- data.frame(y = c(1, 0, 1, 0, 1), x = c(0, 1, 2, 3, 4))
sample_b <- data.frame(x = c(0.5, 1.5, 2.5, 3.5), w = c(4, 4, 4, 4))
sample_b_svy <- svydesign(ids = ~1, weights = ~w, data = sample_b)

res_pmm <- method_pmm(
  y_nons = sample_a$y,
  X_nons = model.matrix(~x, sample_a),
  X_rand = model.matrix(~x, sample_b),
  svydesign = sample_b_svy,
  control_outcome = control_out(k = 1),
  se = FALSE
)

res_pmm
```

`method_ps` *Propensity Score Model Functions*

Description

Function to specify the propensity score (PS) model for the inverse probability weighting estimator. This function provides basic functions logistic regression with a given link function (currently we support `logit`, `probit` and `cloglog`) with additional information about the analytic variance estimator of the mean.

This is a function returns a `list` of functions that refer to specific estimation methods and variance estimators when whether the IPW alone or the DR estimator is applied. The export of this function is mainly because the functions are used in the variable selection algorithms.

Functions starting with `make_log_like`, `make_gradient` and `make_hessian` refer to the maximum likelihood estimation as described in the Chen et al. (2020) paper. These functions take into account different link functions defined through the `link` argument.

Functions `make_link`, `make_link_inv`, `make_link_der`, `make_link_inv_der`, `make_link_inv_rev`, and `make_link_inv_rev_der` refer to specific link functions and are used in the estimation process.

Functions `variance_covariance1` and `variance_covariance2` refer to the variance estimator of the IPW estimator as defined by Chen et al. (2020).

Functions `b_vec_ipw`, `b_vec_dr` and `t_vec` are specific functions defined in the Chen et al. (2020) that are used in the variance estimator of the IPW or the DR.

Finally, `var_nonprob` is the non-probability component of the DR estimator as defined by Chen et al. (2020).

Usage

```
method_ps(link = c("logit", "probit", "cloglog"), ...)
```

Arguments

`link`	link for the PS model
`...`	Additional, optional arguments.

Value

A `list` of functions and elements for a specific link function with the following entries:

make_log_like log-likelihood function for a specific link function

make_gradient gradient of the loglik

make_hessian hessian of the loglik

make_link link function

make_link_inv inverse link function

make_link_der first derivative of the link function

make_link_inv_der first derivative of the the inverse link function

make_link_inv_rev defines 1/inv_link

make_link_inv_rev_der first derivative of 1/inv_link

variance_covariance1 for the IPW estimator: variance component for the non-probability sample

variance_covariance2 for the IPW estimator: variance component for the probability sample

b_vec_ipw for the IPW estimator: the b function as defined in the Chen et al. (2020, sec. 3.2, eq. (9)-(10); sec 4.1)

b_vec_dr for the DR estimator: the b function as defined in the Chen et al. (2020, sec. 3.3., eq. (14); sec 4.1)

t_vec for the DR estimator: the b function as defined in the Chen et al. (2020, sec. 3.3., eq. (14); sec 4.1)

var_nonprob for the DR estimator: non-probability component of the variance for DR estimator

link name of the selected link function for the PS model (character)

model model type (character)

Author(s)

Łukasz Chrostowski, Maciej Beręsewicz

Examples

```
# Printing information on the model selected

method_ps()

# extracting specific field

method_ps("cloglog")$make_gradient
```

`nobs.nonprob` *Returns the Number of Rows in Samples*

Description

Returns information on the number of rows of the probability sample (if provided) and non-probability sample.

Usage

```
## S3 method for class 'nonprob'
nobs(object, ...)
```

Arguments

object	a nonprob class object
...	other arguments passed to methods (currently not supported)

Value

a named vector with row numbers

---

nonprob *Inference with Non-Probability Survey Samples*

Description

nonprob function provides an access to the various methods for inference based on non-probability surveys (including big data). The function allows to estimate the population mean based on the access to a reference probability sample (via the survey package), as well as totals or 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 (if a reference probability sample is provided).

It provides various inverse probability weighting (e.g. with calibration constraints), mass imputation (e.g. nearest neighbour, predictive mean matching) and doubly robust estimators (e.g. that take into account minimisation of the asymptotic bias of the population mean estimators).

The package uses the survey package functionality when a probability sample is available.

All optional parameters are set to NULL. The obligatory ones include data as well as one of the following three: selection, outcome, or target – depending on which method has been selected. In the case of outcome and target multiple y variables can be specified.

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", "npar"),
  family_outcome = c("gaussian", "binomial", "poisson"),
  subset = NULL,
  strata = NULL,
  case_weights = NULL,
```

```
  na_action = na.omit,
  control_selection = control_sel(),
  control_outcome = control_out(),
  control_inference = control_inf(),
  start_selection = NULL,
  start_outcome = NULL,
  verbose = FALSE,
  se = TRUE,
  ...
)
```

Arguments

data	a data.frame with dataset containing the non-probability sample
selection	a formula (default NULL) for the selection (propensity) score model
outcome	a formula (default NULL) for the outcome (target) model
target	a formula (default NULL) with target variable(s). We allow multiple target variables (e.g. ~y1 + y2 + y3)
svydesign	an optional svydesign2 class object containing a probability sample and design weights. If finite population correction should affect survey-side variance estimates, include the fpc in this object.
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 must be supplied when pop_means is used
pop_size	an optional double value with population size. If omitted when a probability sample is supplied, the survey-weight denominator defaults to sum(weights(svydesign)). Supplying pop_size fixes the population-size denominator used by known-N estimators such as IPW-MLE; it does not add or modify finite population correction in svydesign. For IPW-MLE without a fixed pop_size, pop_totals, or pop_means, the point estimator is Hajek-type, with the estimated IPW-weight total as its denominator. For IPW-GEE with svydesign, the point estimator uses the survey-weight denominator sum(weights(svydesign)); if a supplied pop_size differs from that denominator, a warning is issued.
method_selection	a character (default logit) indicating the method for the propensity score link function.
method_outcome	a character (default glm) indicating the method for the outcome model.
family_outcome	a character (default gaussian) describing the error distribution and the link function to be used in the model. Currently supports: gaussian with the 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 (not yet supported, for further development)

`case_weights` an optional positive, finite `numeric vector` of prior weights to be used in the fitting process. It is assumed that this vector contains frequency or analytic weights (i.e. rows of the `data` argument are repeated according to the values of the `case_weights` argument), not probability/design weights.

`na_action` a function which indicates what should happen when the data contain `NAs` (default `na.omit` and it is the only method currently supported)

`control_selection` a `list` (default `control_sel()` result) indicating parameters to be used when fitting the selection model for propensity scores. To change the parameters one should use the `control_sel()` function

`control_outcome` a `list` (default `control_out()` result) indicating parameters to be used when fitting the model for the outcome variable. To change the parameters one should use the `control_out()` function

`control_inference` a `list` (default `control_inf()` result) indicating parameters to be used for inference based on probability and non-probability samples. To change the parameters one should use the `control_inf()` function

`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` a logical value (default `FALSE`) whether detailed information on the fitting should be presented

`se` Logical value (default `TRUE`) indicating whether to calculate and return standard error of estimated mean

`...` Additional, optional arguments (not yet supported)

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_{NP} be the non-probability sample with the design matrix of covariates as

$$\boldsymbol{X}_{\text{NP}} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n_{\text{NP}1}} & x_{n_{\text{NP}2}} & \cdots & x_{n_{\text{NP}p}} \end{bmatrix},$$

and vector of outcome variable

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n_{\text{NP}}} \end{bmatrix}.$$

On the other hand, let S_{P} be the probability sample with design matrix of covariates be

$$\boldsymbol{X}_{\mathrm{P}} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n_{\mathrm{P}}1} & x_{n_{\mathrm{P}}2} & \cdots & x_{n_{\mathrm{P}}p} \end{bmatrix}.$$

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_{P}. The implemented estimators assume that outcome values are observed for the non-probability sample S_{NP}; outcome values observed in the probability sample S_{P} are not used. Linked overlap handling for units appearing in both samples is not currently implemented; `control_sel()` arguments `dependence` and `key` are placeholders for future development.

Supported outcome types depend on the estimator family:

- IPW: numeric targets whose population mean is meaningful, including continuous, count, and 0/1 binary variables. No outcome model is fitted.
- Mass imputation with `method_outcome = "glm"`: continuous, count, or binary variables through `family_outcome = "gaussian"`, `"poisson"`, or `"binomial"`.
- Mass imputation with `method_outcome = "nn"`, `"pmm"`, or `"npar"`: numeric targets; categorical, ordinal, survival, and other structured outcomes are not supported.
- Doubly robust: GLM outcome models only; use `family_outcome = "gaussian"`, `"poisson"`, or `"binomial"`.

In general we make the following assumptions:

1. The selection indicator of belonging to non-probability sample $I_{\mathrm{NP},i}$ 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., $\pi_{\mathrm{NP},i} > 0$ for all i.
3. The indicator variables $I_{\mathrm{NP},i}$ and $I_{\mathrm{NP},j}$ 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. With inverse probability weights $\hat{d}_{\mathrm{NP},i} = 1/\hat{\pi}_{\mathrm{NP},i}$, the package supports two IPW point-estimator families. The Horvitz-Thompson-type estimator uses an external denominator N_0,

$$\hat{\mu}_{IPW,HT} = \frac{1}{N_0} \sum_{i\in S_{\mathrm{NP}}} w_i \hat{d}_{\mathrm{NP},i} y_i,$$

   where w_i are optional `case_weights`. The Hajek-type estimator uses the estimated IPW total as the denominator,

$$\hat{\mu}_{IPW,H} = \frac{\sum_{i\in S_{\mathrm{NP}}} w_i \hat{d}_{\mathrm{NP},i} y_i}{\sum_{i\in S_{\mathrm{NP}}} w_i \hat{d}_{\mathrm{NP},i}}.$$

For IPW-MLE, omitting a fixed population size (`pop_size`, `pop_totals`, or `pop_means` with `pop_size`) gives the Hajek-type estimator. Supplying a fixed population size or population totals gives the Horvitz-Thompson-type estimator. For IPW-GEE with a reference probability sample, the denominator is `sum(weights(svydesign))`; `gee_h_fun` changes the selection-model estimating equation, not the point-estimator family. For IPW-GEE with population totals or means, the denominator comes from those totals. 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

$$\ell^*(\boldsymbol{\theta}) = \sum_{i \in S_{\mathrm{NP}}} \log\left\{\frac{\pi(\boldsymbol{x}_i, \boldsymbol{\theta})}{1-\pi(\boldsymbol{x}_i, \boldsymbol{\theta})}\right\} + \sum_{i \in S_{\mathrm{P}}} d_{\mathrm{P},i} \log\left\{1-\pi(\boldsymbol{x}_i, \boldsymbol{\theta})\right\}.$$

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.

$$\mathbf{U}(\boldsymbol{\theta}) = \sum_{i \in S_{\mathrm{NP}}} \mathbf{h}\left(\mathbf{x}_i, \boldsymbol{\theta}\right) - \sum_{i \in S_{\mathrm{P}}} d_{\mathrm{P},i} \pi\left(\mathbf{x}_i, \boldsymbol{\theta}\right) \mathbf{h}\left(\mathbf{x}_i, \boldsymbol{\theta}\right).$$

Notice that for $\mathbf{h}\left(\mathbf{x}_i, \boldsymbol{\theta}\right) = \frac{\boldsymbol{x}}{\pi(\boldsymbol{x}, \boldsymbol{\theta})}$ 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:

$$\hat{\mu}_{MI} = \frac{1}{N_{\mathrm{P}}} \sum_{i \in S_{\mathrm{P}}} d_{\mathrm{P},i} \hat{y}_i.$$

It opens 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

$$\hat{\mu}_{DR} = \frac{1}{N_{\mathrm{NP}}} \sum_{i \in S_{\mathrm{NP}}} \hat{d}_{\mathrm{NP},i}(y_i - \hat{y}_i) + \frac{1}{N_{\mathrm{P}}} \sum_{i \in S_{\mathrm{P}}} d_{\mathrm{P},i} \hat{y}_i.$$

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

$$\begin{aligned} bias(\hat{\mu}_{DR}) =& \mathbb{E}(\hat{\mu}_{DR} - \mu) \\ =& \mathbb{E}\left\{\frac{1}{N}\sum_{i=1}^{N}\left(\frac{I_{\mathrm{NP},i}}{\pi_{\mathrm{NP},i}(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\theta})} - 1\right)(y_i - \mathrm{m}(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta}))\right\} \\ &+ \mathbb{E}\left\{\frac{1}{N}\sum_{i=1}^{N}\left(I_{\mathrm{P},i} d_{\mathrm{P},i} - 1\right)\mathrm{m}(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta})\right\}, \end{aligned}$$

lead us to system of equations

$$J(\theta,\beta) = \left\{ \begin{array}{c} J_1(\theta,\beta) \\ J_2(\theta,\beta) \end{array} \right\} = \left\{ \begin{array}{c} \sum_{i=1}^{N} I_{\text{NP},i} \left\{ \frac{1}{\pi(\boldsymbol{x}_i,\boldsymbol{\theta})} - 1 \right\} \{y_i - m(\boldsymbol{x}_i,\boldsymbol{\beta})\} \boldsymbol{x}_i \\ \sum_{i=1}^{N} \frac{I_{\text{NP},i}}{\pi(\boldsymbol{x}_i,\boldsymbol{\theta})} \frac{\partial m(\boldsymbol{x}_i,\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} - \sum_{i \in \mathcal{S}_{\text{P}}} d_{\text{P},i} \frac{\partial m(\boldsymbol{x}_i,\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} \end{array} \right\},$$

where $m(\boldsymbol{x}_i, \boldsymbol{\beta})$ is a mass imputation (regression) model for the outcome variable and propensity scores $\pi_{\text{NP},i}$ 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, the bootstrap approach can be used for variance estimation.

The doubly robust analytic (Taylor-linearisation) variance is derived under the logistic propensity model (Chen, Li & Wu 2020, Theorem 2). For the `probit` and `cloglog` selection links the probability-sample (design) variance component is a conservative approximation: it tends to overestimate the standard error and can be numerically unstable when fitted propensities approach 1. For doubly robust inference with those links, `control_inf(var_method = "bootstrap")` is recommended.

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

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

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 `control_inf()` function to `TRUE`. In addition, in the other control functions such as `control_sel()` and `control_out()` 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 the `nonprob` class (it is actually a `list`) which contains the following elements:

- `call` – the call of the `nonprob` function
- `data` – a `data.frame` passed from the `nonprob` function `data` argument
- `X` – a `model.matrix` containing data from probability (first $n_{S_{\text{P}}}$ rows) and non-probability samples (next $n_{S_{\text{P}}}$ rows) if specified at a function call
- `y` – a `list` of vector of outcome variables if specified at a function call
- `R` – a `numeric vector` indicating whether a unit belongs to the probability (0) or non-probability (1) units in the matrix X

- ps_scores – a numeric vector of estimated propensity scores for probability and non-probability sample
- case_weights – a vector of case weights for non-probability sample based on the call
- ipw_weights – a vector of inverse probability weights for non-probability sample (if applicable)
- boot_ipw_weights – a matrix of bootstrap inverse probability weights for the non-probability sample when bootstrap variance estimation is used and bootstrap results are kept (if applicable)
- control – a list of control functions based on the call
- output – a data.frame with the estimated means and standard errors for the variables specified in the target or outcome arguments
- SE – a data.frame with standard error of the estimator of the population mean, divided into errors from probability and non-probability samples (if applicable)
- confidence_interval – a data.frame with confidence interval of population mean estimator
- nonprob_size – a scalar numeric vector denoting the size of non-probability sample
- prob_size – a scalar numeric vector denoting the size of probability sample
- pop_size – a scalar numeric vector estimated population size derived from estimated weights (non-probability sample) or known design weights (probability sample)
- pop_size_fixed – a logical value whether the population size was fixed (known) or estimated (unknown)
- ipw_estimator – a character value with the IPW point-estimator family ("ht" or "hajek", if applicable)
- ipw_denominator – a scalar numeric vector with the denominator used for the IPW point estimator (if applicable)
- ipw_denominator_source – a character value describing the source of the IPW point-estimator denominator (if applicable)
- pop_totals – a numeric vector with the total values of the auxiliary variables derived from a probability sample or based on the call
- pop_means – a numeric vector with the mean values of the auxiliary variables derived from a probability sample or based on the call
- outcome – a list containing information about the fitting of the mass imputation model. Structure of the object is based on the method_outcome and family_outcome arguments which point to specific methods as defined by functions method_* (if specified in the call)
- selection – a list containing information about the fitting of the propensity score model. Structure of the object is based on the method_selection argument which point to specific methods as defined by functions method_ps (if specified in the call)
- boot_sample – a matrix with bootstrap estimates of the target variable(s) (if specified)
- svydesign – a svydesign2 object (if specified)
- ys_rand_pred – a list of predicted values for the target variable(s) for the probability sample (for the MI and DR estimator)

- ys_nons_pred – a list of predicted values for the target variable(s) for the non-probability sample (for the MI and DR estimator)
- ys_resid – a list of residuals for the target variable(s) for the non-probability sample (for the MI and DR estimator)
- estimator – a character vector with information what type of estimator was selected (one of c("ipw", "mi", "dr")).
- selection_formula – a formula based on the selection argument (if specified)
- estimator_method – a character vector with information on the detailed method applied (for the print method)

Author(s)

Łukasz Chrostowski, Maciej Beręsewicz, Piotr Chlebicki

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.

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

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

control_inf() – 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
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 <- n_b* z_prob/sum(z_prob)

# data
sim_data$flag_nonprob <- as.numeric(runif(N) < rho) ## sampling nonprob
sim_data$flag_prob <- as.numeric(runif(n_b) < 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 = y30 + y50 + y80 ~ x1 + x2 + x3 + x4,
  data = nonprob_df,
```

```
  svydesign = svyprob
)
mi_res
## inverse probability weighted estimator
ipw_res <- nonprob(
  selection = ~ x1 + x2 + x3 + x4,
  target = ~y30 + y50 + y80,
  data = nonprob_df,
  svydesign = svyprob
)
ipw_res
## doubly robust estimator
dr_res <- nonprob(
  outcome = y30 + y50 + y80 ~ x1 + x2 + x3 + x4,
  selection = ~ x1 + x2 + x3 + x4,
  data = nonprob_df,
  svydesign = svyprob
)
dr_res
```

`plot.nonprob` *Plots the Estimated Mean(s) and Their Confidence Interval(s)*

Description

Simple plotting method that compares the estimated mean(s) and CI(s) with the naive (uncorrected) estimates.

Usage

```
## S3 method for class 'nonprob'
plot(x, ...)
```

Arguments

`x`	the `nonprob` class object
`...`	other arguments passed to the plot method (currently not supported)

Examples

```
data(admin)
data(jvs)

jvs_svy <- svydesign(ids = ~ 1,  weights = ~ weight,
strata = ~ size + nace + region, data = jvs)

ipw_est1 <- nonprob(selection = ~ region + private + nace + size,
target = ~ single_shift,
```

```
svydesign = jvs_svy,
data = admin, method_selection = "logit")

plot(ipw_est1)
```

---

pop_size *Returns Population Size (Estimated or Fixed)*

Description

Returns population size that is assumed to be

- `fixed` – if it is based on the `pop_size` argument or population totals,
- `estimated` – if it is based on the probability survey specified in the `svydesign` or, for Hajek-type IPW-MLE, on the estimated IPW total for the non-probability sample.

Usage

```
pop_size(object)
```

Arguments

`object`	object returned by the `nonprob` function.

Value

a scalar returning the value of the population size.

Examples

```
sample_a <- data.frame(y = c(1, 0, 1, 0, 1), x = c(0, 1, 2, 3, 4))
sample_b <- data.frame(x = c(0.5, 1.5, 2.5, 3.5), w = c(4, 4, 4, 4))
sample_b_svy <- svydesign(ids = ~1, weights = ~w, data = sample_b)

ipw_est1 <- nonprob(
  selection = ~x,
  target = ~y,
  svydesign = sample_b_svy,
  data = sample_a,
  method_selection = "logit",
  se = FALSE
)

ipw_est2 <- nonprob(
  selection = ~x,
  target = ~y,
  svydesign = sample_b_svy,
  data = sample_a,
  method_selection = "logit",
```

```
  control_selection = control_sel(est_method = "gee", gee_h_fun = 1),
  se = FALSE
)

## estimated population size based on the non-calibrated IPW (MLE)
pop_size(ipw_est1)

## estimated population size based on the calibrated IPW (GEE)
pop_size(ipw_est2)
```

`print.nonprob_summary` *Print Method for the Nonprob_summary Object*

Description

Print method for the `nonprob_summary` object which allows for specification what should be printed or not.

Usage

```
## S3 method for class 'nonprob_summary'
print(x, resid = TRUE, pred = TRUE, digits = 4, ...)
```

Arguments

`x`	a `nonprob` object
`resid`	whether distribution of residuals should be printed (default is `TRUE`)
`pred`	whether distribution of predictions should be printed (default is `TRUE`)
`digits`	number of digits to be printed (default 4)
`...`	further parameters passed to the print method (currently not supported)

`summary.nonprob` *Summary Statistics for Models of the Nonprob Class*

Description

Summarises the `nonprob` class object. The summary depends on the type of the estimator (i.e. IPW, MI, DR)

Usage

```
## S3 method for class 'nonprob'
summary(object, ...)
```

Arguments

object	object of the nonprob class
...	Additional optional arguments

Value

An object of nonprob_summary class containing:

- call call
- estimator type of estimator
- control list of controls
- ipw_weights estimated IPW weights
- ipw_weights_total estimated IPW total (sum)
- ps_scores_nonprob estimated propensity scores for non-probability sample
- ps_scores_prob estimated propensity scores for probability sample
- case_weights case weights
- output estimated means and standard errors
- SE estimated standard errors of V1 and V2
- confidence_interval confidence intervals
- nonprob_size size of the non-probability sample
- prob_size size of the probability sample
- pop_size population size
- pop_size_fixed whether the population size is treated as fixed
- ipw_estimator IPW point-estimator family ("ht" or "hajek")
- ipw_denominator denominator used for the IPW point estimator
- ipw_denominator_source source of the IPW point-estimator denominator
- no_prob whether probability sample was provided
- outcome model details
- selection selection details
- estimator_method estimator method
- selection_formula selection formula
- outcome_formula outcome formula
- vars_selection whether variable selection algorithm was applied
- vars_outcome variables of the outcome models
- ys_rand_pred predicted values for the random sample (if applies)
- ys_nons_pred predicted values for the non-probability sample
- ys_resid residuals for the non-probability sample

Examples

```
data(admin)
data(jvs)

jvs_svy <- svydesign(ids = ~ 1,  weights = ~ weight,
strata = ~ size + nace + region, data = jvs)

ipw_est1 <- nonprob(selection = ~ region + private + nace + size,
target = ~ single_shift,
svydesign = jvs_svy,
data = admin, method_selection = "logit"
)
summary(ipw_est1)
```

`update.nonprob` *Update Method for the Nonprob Object with Changed Arguments or Parameters*

Description

The `update` method for the `nonprob` class object that allows to re-estimate a given model with changed parameters. This is in particular useful if a user would like to change method or estimate standard errors if they were not estimated in the first place.

Usage

```
## S3 method for class 'nonprob'
update(object, ..., evaluate = TRUE)
```

Arguments

`object`	the `nonprob` class object
`...`	arguments passed to the `nonprob` class object
`evaluate`	If true evaluate the new call else return the call

Value

returns a `nonprob` object

Author(s)

Maciej Beręsewicz

Examples

```
data(admin)
data(jvs)

jvs_svy <- svydesign(ids = ~ 1,  weights = ~ weight,
strata = ~ size + nace + region, data = jvs)

ipw_est1 <- nonprob(selection = ~ region + private + nace + size,
target = ~ single_shift,
svydesign = jvs_svy,
data = admin, method_selection = "logit", se = FALSE
)

ipw_est1

update(ipw_est1, se = TRUE)
```

---

weights.nonprob *Extracts the Inverse Probability (Propensity Score) Weights*

Description

A generic function `weights` that returns inverse probability weights (if present)

Usage

```
## S3 method for class 'nonprob'
weights(object, ...)
```

Arguments

`object`	a `nonprob` class object
`...`	other arguments passed to methods (currently not supported)

Value

A vector of weights or a `NULL` extracted from the `nonprob` object i.e. element `"ipw_weights"`

Examples

```
data(admin)
data(jvs)

jvs_svy <- svydesign(ids = ~ 1,  weights = ~ weight,
strata = ~ size + nace + region, data = jvs)

ipw_est1 <- nonprob(selection = ~ region + private + nace + size,
target = ~ single_shift,
```

```
svydesign = jvs_svy,
data = admin, method_selection = "logit", se = FALSE
)

summary(weights(ipw_est1))
```

Index

∗ **datasets**
admin, 3
jvs, 12

admin, 3

check_balance, 3
coef.nonprob, 4
confint.nonprob, 5
control_inf, 6
control_inf(), *22*, *32*, *34*
control_out, 8
control_out(), *22*, *32*, *34*
control_sel, 10
control_sel(), *32*, *34*

extract, 12

jvs, 12

maxLik::maxLik(), *10*, *34*
method_glm, 13
method_nn, 16
method_npar, 19
method_pmm, 21
method_ps, 25

ncvreg::cv.ncvreg(), *34*
nleqslv::nleqslv(), *11*, *34*
nobs.nonprob, 26
nonprob, 27
nonprob(), *7*, *9*, *11*

plot.nonprob, 36
pop_size, 37
print.nonprob_summary, 38

RANN::nn2, *16*, *22*
RANN::nn2(), *8, 9*, *17*, *31*, *34*

stats::glm(), *31*, *34*
stats::loess, *9*
stats::loess.control, *9*
stats::optim(), *10*, *34*
summary.nonprob, 38
survey::as.svrepdesign(), *7*

update.nonprob, 40

weights.nonprob, 41