Package 'NMAR'

January 16, 2026

Type Package

Title Estimation under not Missing at Random Nonresponse

Version 0.1.1

Description Methods to estimate finite-population parameters under nonresponse that is not missing at random (NMAR, nonignorable). Incorporates auxiliary information and user-specified response models,
and supports independent samples and complex survey designs via objects from the 'survey' package.
Provides diagnostics and optional variance estimates. For methodological background see Qin, Leung and Shao (2002) <doi:10.1198/016214502753479338> and Riddles, Kim and Im (2016) <doi:10.1093/jssam/smv047>.

License MIT + file LICENSE

URL https://github.com/ncn-foreigners/NMAR,
https://ncn-foreigners.ue.poznan.pl/NMAR/index.html

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

Encoding UTF-8

Imports stats, nleqslv, utils, generics, Formula

RoxygenNote 7.3.3

Suggests knitr, rmarkdown, testthat (>= 3.0.0), numDeriv, survey, svrep, broom, progressr, future, future.apply, spelling

VignetteBuilder knitr

Config/testthat/edition 3

Depends R (>= 3.5)

LazyData true

Language en-US

NeedsCompilation no

Author Maciej Beresewicz [aut, cre] (ORCID: <https://orcid.org/0000-0002-8281-4301>),
Igor Kołodziej [aut, ctb],
Mateusz Iwaniuk [aut, ctb]

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

Repository CRAN

Date/Publication 2026-01-16 10:50:02 UTC

Contents

coef.nmar_result *Default coefficients for NMAR results*

Description

Returns missingness-model coefficients if available.

Usage

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

Arguments

object	An ‘nmar_result‘ object.
...	Ignored.

Value

A named numeric vector or ‘NULL‘.

coef.summary_nmar_result *Coefficient table for summary objects*

Description

Returns a coefficients table (Estimate, Std. Error, statistic, p-value) from a ‘summary_nmar_result*‘ object when missingness-model coefficients and a variance matrix are available. If the summary does not carry missingness-model coefficients, returns ‘NULL‘.

Usage

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

Arguments

object	An object of class ‘summary_nmar_result‘ (or subclass).
...	Ignored.

Details

The statistic column is labelled "t value" when finite degrees of freedom are available (e.g., survey designs); otherwise, it is labelled "z value".

Value

A data.frame with rows named by coefficient, or ‘NULL‘ if not available.

`confint.nmar_result` *Wald confidence interval for base NMAR results*

Description

Wald confidence interval for base NMAR results

Usage

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

Arguments

`object`	An object of class ‘nmar_result‘.
`parm`	Ignored.
`level`	Confidence level.
`...`	Ignored.

Value

A 1x2 numeric matrix with confidence limits.

`confint.summary_nmar_result`
Confidence intervals for coefficient table (summary objects)

Description

Returns Wald-style confidence intervals for missingness-model coefficients from a ‘summary_nmar_result*‘ object. Uses t-quantiles when finite degrees of freedom are available, otherwise normal quantiles.

Usage

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

Arguments

object	An object of class 'summary_nmar_result' (or subclass).
parm	A specification of which coefficients are to be given confidence intervals, either a vector of names or a vector of indices; by default, all coefficients are considered.
level	The confidence level required.
...	Ignored.

Value

A numeric matrix with columns giving lower and upper confidence limits for each parameter. Row names correspond to coefficient names. Returns 'NULL' if coefficients are unavailable.

---

el_engine *Empirical likelihood (EL) engine for NMAR*

Description

Constructs an engine specification for the empirical likelihood (EL) estimator of a full-data mean under nonignorable nonresponse (NMAR).

Usage

```
el_engine(
  standardize = TRUE,
  trim_cap = Inf,
  on_failure = c("return", "error"),
  variance_method = c("bootstrap", "none"),
  bootstrap_reps = 500,
  auxiliary_means = NULL,
  control = list(),
  strata_augmentation = TRUE,
  n_total = NULL,
  start = NULL,
  family = c("logit", "probit")
)
```

Arguments

standardize	logical; standardize predictors. Default TRUE.
trim_cap	numeric; cap for EL weights (Inf = no trimming).
on_failure	character; "return" or "error" on solver failure.
variance_method	character; one of "bootstrap" or "none".
bootstrap_reps	integer; number of bootstrap replicates when variance_method = "bootstrap".

`auxiliary_means`
: named numeric vector; population means for auxiliary design columns. Names must match the materialized model.matrix column names on the first RHS (after formula expansion), e.g., factor indicator columns created by `model.matrix()` or transformed terms like `I(X^2)`. Auxiliary intercepts are always dropped automatically, so do not supply `(Intercept)`. If `NULL` (default) and the outcome contains at least one `NA`, auxiliary means are estimated from the full input (including nonrespondents): IID uses unweighted column means of the auxiliary design; survey designs use the design-weighted means based on `weights(design)`. This corresponds to the QLS case where μ_x is replaced by $\bar{X}$ (the full-sample mean) when auxiliary variables are observed for all sampled units.

`control`
: Optional solver configuration forwarded to `nleqslv::nleqslv()`. Provide a single list that may include solver tolerances (e.g., `xtol`, `ftol`, `maxit`) and, optionally, top-level entries `global` and `xscalm` for globalization and scaling. Example: `control = list(maxit = 500, xtol = 1e-10, ftol = 1e-10, global = "qline", xscalm = "auto")`.

`strata_augmentation`
: logical; when `TRUE` (default), survey designs with an identifiable strata structure are augmented with stratum indicators and corresponding population shares in the auxiliary block (Wu-style strata augmentation). Has no effect for `data.frame` inputs or survey designs without strata.

`n_total`
: numeric; optional when supplying respondents-only data (no `NA` in the outcome). For `data.frame` inputs, set to the total number of sampled units before filtering to respondents. For `survey.design` inputs, set to the total design-weight total on the same analysis scale as `weights(design)` (default `sum(weights(design))`). If omitted and the outcome contains no NAs, the estimator errors, requesting `n_total`.

`start`
: list; optional starting point for the solver. Fields:

  - `beta`: named numeric vector of missingness-model coefficients on the original (unscaled) scale, including `(Intercept)`.
  - `W` or `z`: starting value for population response rate (`0 < W < 1`) or its logit (`z`). If both are provided, `z` takes precedence.
  - `lambda`: named numeric vector of auxiliary multipliers on the original scale (names must match auxiliary design columns; no intercept). Values are mapped to the scaled space internally.

`family`
: Missingness (response) model family. Either `"logit"` (default) or `"probit"`, or a custom family object: a list with components `name`, `linkinv`, `mu.eta`, `score_eta`, and optionally `d2mu.deta2`. When `d2mu.deta2` is absent the solver uses Broyden/numeric Jacobians.

Details

The implementation follows Qin, Leung, and Shao (2002): the response mechanism is modeled as $w(y, x; \beta) = P(R = 1 \mid Y = y, X = x)$ and the joint law of (Y, X) is represented nonparametrically by respondent masses that satisfy empirical likelihood constraints. The mean is estimated as a respondent weighted mean with weights proportional to $\tilde{w}_i = a_i / D_i(\beta, W, \lambda)$, where a_i are base weights ($a_i \equiv 1$ for IID data and $a_i = d_i$ for survey designs) and D_i is the EL denominator.

For `data.frame` inputs the estimator solves the Qin-Leung-Shao (QLS) estimating equations for (β, W, λ_x) with W reparameterized as $z = \text{logit}(W)$, and profiles out the response multiplier λ_W using the closed-form QLS identity (their Eq. 10). For `survey.design` inputs the estimator uses a design-weighted analogue (Chen and Sitter 1999; Wu 2005) with an explicit λ_W and an additional linkage equation involving the nonrespondent design-weight total T_0.

Numerical stability:

- W is optimized on the logit scale so $0 < W < 1$.
- The response-model linear predictor is capped and EL denominators D_i are floored at a small positive value; the analytic Jacobian is consistent with this guard via an active-set mask.
- Optional trimming (`trim_cap`) is applied only after solving, to the unnormalized masses $\tilde{w}_i = a_i/D_i$; this changes the returned weights and therefore the point estimate.

Formula syntax and data constraints: `nmar()` accepts a partitioned right-hand side `y_miss ~ auxiliaries | response_only`. Variables left of `|` enter auxiliary moment constraints; variables right of `|` enter only the response model. The outcome (LHS) is always included as a response-model predictor through the evaluated LHS expression; explicit use of the outcome on the RHS is rejected. The response model always includes an intercept; the auxiliary block never includes an intercept.

To include a covariate in both the auxiliary constraints and the response model, repeat it on both sides, e.g. `y_miss ~ X | X`.

Auxiliary means: If `auxiliary_means = NULL` (default) and the outcome contains at least one `NA`, auxiliary means are estimated from the full input and used as $\bar{X}$ in the QLS constraints. For respondents-only data (no `NA` in the outcome), `n_total` must be supplied; and if the auxiliary RHS is non-empty, `auxiliary_means` must also be supplied. When `standardize = TRUE`, supply `auxiliary_means` on the original data scale; the engine applies the same standardization internally.

Survey scale: For `survey.design` inputs, `n_total` (if provided) must be on the same analysis scale as `weights(design)`. The default is `sum(weights(design))`.

Convergence and identification: the stacked EL system can have multiple solutions. Adding response-only predictors (variables to the right of `|`) can make the problem sensitive to starting values. Inspect diagnostics such as `jacobian_condition_number` and consider supplying `start = list(beta = ..., W = ...)` when needed.

Variance: The EL engine supports bootstrap standard errors via `variance_method = "bootstrap"` or can skip variance with `variance_method = "none"`. Set a seed for reproducible bootstrap results.

Bootstrap requires suggested packages: for IID resampling it requires `future.apply` (and `future`); for survey replicate-weight bootstrap it requires `survey` and `svrep`.

Value

A list of class `"nmar_engine_el"` (also inheriting from `"nmar_engine"`) containing configuration fields to be supplied to `nmar()`. Users rarely access fields directly; instead, pass the engine to `nmar()` together with a formula and data.

References

Qin, J., Leung, D., and Shao, J. (2002). Estimation with survey data under nonignorable nonresponse or informative sampling. Journal of the American Statistical Association, 97(457), 193-200. doi:10.1198/016214502753479338

Chen, J., and Sitter, R. R. (1999). A pseudo empirical likelihood approach for the effective use of auxiliary information in complex surveys. Statistica Sinica, 9, 385-406.

Wu, C. (2005). Algorithms and R codes for the pseudo empirical likelihood method in survey sampling. Survey Methodology, 31(2), 239-243.

See Also

nmar, weights.nmar_result, summary.nmar_result

Examples

```
set.seed(1)
n <- 200
X <- rnorm(n)
Y <- 2 + 0.5 * X + rnorm(n)
p <- plogis(-0.7 + 0.4 * scale(Y)[, 1])
R <- runif(n) < p
if (all(R)) R[1] <- FALSE
df <- data.frame(Y_miss = Y, X = X)
df$Y_miss[!R] <- NA_real_

# Estimate auxiliary mean from full data (QLS "use Xbar" case)
eng <- el_engine(auxiliary_means = NULL, variance_method = "none")

# Put X in both the auxiliary block and the response model (QLS-like)
fit <- nmar(Y_miss ~ X | X, data = df, engine = eng)
summary(fit)


# Response-only predictors can be placed to the right of |:
Z <- rnorm(n)
df2 <- data.frame(Y_miss = Y, X = X, Z = Z)
df2$Y_miss[!R] <- NA_real_
eng2 <- el_engine(auxiliary_means = NULL, variance_method = "none")
fit2 <- nmar(Y_miss ~ X | Z, data = df2, engine = eng2)
print(fit2)

# Survey design usage
if (requireNamespace("survey", quietly = TRUE)) {
  des <- survey::svydesign(ids = ~1, weights = ~1, data = df)
  eng3 <- el_engine(auxiliary_means = NULL, variance_method = "none")
  fit3 <- nmar(Y_miss ~ X, data = des, engine = eng3)
  summary(fit3)
}
```

`engine_config` *Extract engine configuration*

Description

Returns the underlying configuration of an engine as a named list. This is intended for programmatic inspection (e.g., parameter tuning, logging). The returned object should be treated as read-only.

Usage

```
engine_config(x)
```

Arguments

x	An object inheriting from class ‘nmar_engine‘.

Value

A named list of configuration fields.

`engine_name` *Canonical engine name*

Description

Returns a stable, machine-friendly identifier for an engine object. This identifier is also used in ‘nmar_result$meta$engine_name‘ to keep a consistent naming scheme between configurations and results.

Usage

```
engine_name(x)
```

Arguments

x	An object inheriting from class ‘nmar_engine‘.

Value

A single character string, e.g. "empirical_likelihood".

`exptilt_engine` *Exponential tilting (ET) engine for NMAR estimation*

Description

Constructs a configuration for the exponential tilting estimator under nonignorable nonresponse (NMAR). The estimator solves $S_2(\phi, \hat{\gamma}) = 0$, using nleqslv to apply EM algorithm.

Usage

```
exptilt_engine(
  standardize = FALSE,
  on_failure = c("return", "error"),
  variance_method = c("bootstrap", "none"),
  bootstrap_reps = 10,
  supress_warnings = FALSE,
  control = list(),
  family = c("logit", "probit"),
  y_dens = c("normal", "lognormal", "exponential", "binomial"),
  stopping_threshold = 1,
  sample_size = 2000
)
```

Arguments

`standardize`	logical; standardize predictors. Default `TRUE`.
`on_failure`	character; `"return"` or `"error"` on solver failure
`variance_method`	character; one of `"bootstrap"`, or `"none"`.
`bootstrap_reps`	integer; number of bootstrap replicates when `variance_method = "bootstrap"`.
`supress_warnings`	Logical; suppress variance-related warnings.
`control`	Named list of control parameters passed to `nleqslv::nleqslv`. Common parameters include: • `maxit`: Maximum number of iterations (default: 100) • `method`: Solver method - `"Newton"` or `"Broyden"` (default: `"Newton"`) • `global`: Global strategy - `"dbldog"`, `"pwldog"`, `"qline"`, `"gline"`, `"hook"`, or `"none"` (default: `"dbldog"`) • `xtol`: Tolerance for relative error in solution (default: 1e-8) • `ftol`: Tolerance for function value (default: 1e-8) • `btol`: Tolerance for backtracking (default: 0.01) • `allowSingular`: Allow singular Jacobians (default: `TRUE`) See `?nleqslv::nleqslv` for full details.

`family` character; response model family, either `"logit"` or `"probit"`, or a family object created by `logit_family()` / `probit_family()`.

`y_dens` Outcome density model (`"auto"`, `"normal"`, `"lognormal"`, `"exponential"`, or `"binomial"`).

`stopping_threshold` Numeric; early stopping threshold. If the maximum absolute value of the score function falls below this threshold, the algorithm stops early (default: 1).

`sample_size` Integer; maximum sample size for stratified random sampling (default: 2000). When the dataset exceeds this size, a stratified random sample is drawn to optimize memory usage. The sampling preserves the ratio of respondents to non-respondents in the original data.

Details

The method is a robust Propensity-Score Adjustment (PSA) approach for Not Missing at Random (NMAR). It uses Maximum Likelihood Estimation (MLE), basing the likelihood on the observed part of the sample ($f(\boldsymbol{Y}_i|\delta_i = 1, \boldsymbol{X}_i)$), making it robust against outcome model misspecification. The propensity score is estimated by assuming an instrumental variable (X_2) that is independent of the response status given other covariates and the study variable. Estimator calculates fractional imputation weights w_i. The final estimator is a weighted average, where the weights are the inverse of the estimated response probabilities $\hat{\pi}_i$, satisfying the estimating equation:

$$\sum_{i \in \mathcal{R}} \frac{\boldsymbol{g}(\boldsymbol{Y}_i, \boldsymbol{X}_i; \boldsymbol{\theta})}{\hat{\pi}_i} = 0,$$

where $\mathcal{R}$ is the set of observed respondents.

Value

An engine object of class `c("nmar_engine_exptilt","nmar_engine")`. This is a configuration list; it is not a fit. Pass it to nmar.

References

Minsun Kim Riddles, Jae Kwang Kim, Jongho Im A Propensity-score-adjustment Method for Non-ignorable Nonresponse *Journal of Survey Statistics and Methodology*, Volume 4, Issue 2, June 2016, Pages 215–245.

Examples

```
generate_test_data <- function(
  n_rows = 500,
  n_cols = 1,
  case = 1,
  x_var = 0.5,
  eps_var = 0.9,
  a = 0.8,
  b = -0.2
) {
# Generate X variables - fixed to match comparison
```

```
  X <- as.data.frame(replicate(n_cols, rnorm(n_rows, 0, sqrt(x_var))))
  colnames(X) <- paste0("x", 1:n_cols)

# Generate Y - fixed coefficients to match comparison
  eps <- rnorm(n_rows, 0, sqrt(eps_var))
  if (case == 1) {
# Use fixed coefficient of 1 for all x variables to match: y = -1 + x1 + epsilon
    X$Y <- as.vector(-1 + as.matrix(X) %*% rep(1, n_cols) + eps)
  }
  else if (case == 2) {
    X$Y <- -2 + 0.5 * exp(as.matrix(X) %*% rep(1, n_cols)) + eps
  }
  else if (case == 3) {
    X$Y <- -1 + sin(2 * as.matrix(X) %*% rep(1, n_cols)) + eps
  }
  else if (case == 4) {
    X$Y <- -1 + 0.4 * as.matrix(X)^3 %*% rep(1, n_cols) + eps
  }

  Y_original <- X$Y

# Missingness mechanism - identical to comparison
  pi_obs <- 1 / (1 + exp(-(a + b * X$Y)))

# Create missing values
  mask <- runif(nrow(X)) > pi_obs
  mask[1] <- FALSE # Ensure at least one observation is not missing
  X$Y[mask] <- NA

  return(list(X = X, Y_original = Y_original))
}
res_test_data <- generate_test_data(n_rows = 500, n_cols = 1, case = 1)
x <- res_test_data$X

exptilt_config <- exptilt_engine(
  y_dens = 'normal',
  control = list(maxit = 10),
  stopping_threshold = 0.1,
  standardize = FALSE,
  family = 'logit',
  bootstrap_reps = 5
)
formula = Y ~ x1
res <- nmar(formula = formula, data = x, engine = exptilt_config, trace_level = 1)
summary(res)
```

`exptilt_nonparam_engine`

Nonparametric exponential tilting (EM) engine for NMAR

Description

Constructs a configuration for the nonparametric exponential tilting estimator under nonignorable nonresponse (NMAR). This engine implements the "Fully Nonparametric Approach" from **Appendix 2** of Riddles et al. (2016). The estimator uses an Expectation-Maximization (EM) algorithm to directly estimate the nonresponse odds $O(x_1, y)$ for aggregated, categorical data.

Usage

```
exptilt_nonparam_engine(refusal_col = "", max_iter = 100, tol_value = 1e-06)
```

Arguments

`refusal_col`	character; the column name in `data` that contains the aggregated counts of non-respondents (refusals).
`max_iter`	integer; the maximum number of iterations for the EM algorithm.
`tol_value`	numeric; the convergence tolerance for the EM algorithm. The loop stops when the sum of absolute changes in the odds matrix is less than this value.

Details

This engine is designed for cases where all variables (outcomes Y, response predictors X_1, and instrumental variables X_2) are categorical, and the input `data` is pre-aggregated into strata.

The method assumes an instrumental variable X_2 is available. The response probability is assumed to depend on X_1 and Y, but *not* on X_2.

The EM algorithm iteratively solves for the nonresponse odds:

$$O^{(t+1)}(x_1^*, y^*) = \frac{M^{(t)}_{y^* x_1^*}}{N_{y^* x_1^*}}$$

where $M^{(t)}_{y^* x_1^*}$ is the expected count of non-respondents (calculated in the E-step) and $N_{y^* x_1^*}$ is the observed count of respondents for a given stratum (x_1, y).

The final output from the `nmar` call is an object containing `data_to_return`, an aggregated data frame where the original 'refusal' counts have been redistributed into the outcome columns (e.g., 'Voted_A', 'Voted_B') as expected non-respondent counts.

Value

An engine object of class `c("nmar_engine_exptilt_nonparam","nmar_engine")`. This is a configuration list; it is not a fit. Pass it to nmar.

References

Minsun Kim Riddles, Jae Kwang Kim, Jongho Im A Propensity-score-adjustment Method for Nonignorable Nonresponse *Journal of Survey Statistics and Methodology*, Volume 4, Issue 2, June 2016, Pages 215–245. (See **Appendix 2** for this specific method).

Examples

```
# Test data (Riddles 2016, Table 9)
voting_data_example <- data.frame(
 Gender = rep(c("Male", "Male", "Male", "Male", "Female", "Female", "Female", "Female"), 1),
  Age_group = c("20-29", "30-39", "40-49", ">=50", "20-29", "30-39", "40-49", "50+"),
  Voted_A = c(93, 104, 146, 560, 106, 129, 170, 501),
  Voted_B = c(115, 233, 295, 350, 159, 242, 262, 218),
  Other = c(4, 8, 5, 3, 8, 5, 5, 7),
  Refusal = c(28, 82, 49, 174, 62, 70, 69, 211),
  Total = c(240, 427, 495, 1087, 335, 446, 506, 937)
)

np_em_config <- exptilt_nonparam_engine(
  refusal_col = "Refusal",
  max_iter = 100,
  tol_value = 0.001
)

# Formula: Y1 + Y2 + ... ~ X1_vars | X2_vars
# Here, Y = Voted_A, Voted_B, Other
#      x1 = Gender (response model)
#      x2 = Age_group (instrumental variable)
em_formula <- Voted_A + Voted_B + Other ~ Gender | Age_group


results_em_np <- nmar(
  formula = em_formula,
  data = voting_data_example,
  engine = np_em_config,
  trace_level = 0
)

# View the final adjusted counts
# (Original counts + expected non-respondent counts)
print(results_em_np$data_final)
```

---

fitted.nmar_result *Default fitted values for NMAR results*

Description

Returns fitted response probabilities if available.

Usage

```
## S3 method for class 'nmar_result'
fitted(object, ...)
```

Arguments

`object`	An ‘nmar_result‘ object.
`...`	Ignored.

Value

A numeric vector (possibly length 0).

`format.nmar_engine` *One-line formatter for NMAR engines*

Description

Returns a single concise line summarizing an engine configuration.

Usage

```
## S3 method for class 'nmar_engine'
format(x, ...)
```

Arguments

`x`	An engine object inheriting from ‘nmar_engine‘.
`...`	Unused.

Value

A length-1 character vector.

`formula.nmar_result` *Default formula for NMAR results*

Description

Returns the estimation formula if available.

Usage

```
## S3 method for class 'nmar_result'
formula(x, ...)
```

Arguments

`x`	An ‘nmar_result‘ object.
`...`	Ignored.

Value

A formula or ‘NULL‘.

`glance.nmar_result` *Glance summary for NMAR results*

Description

One-row diagnostics for NMAR fits.

Usage

```
## S3 method for class 'nmar_result'
glance(x, ...)
```

Arguments

`x`	An object of class ‘nmar_result‘.
`...`	Ignored.

Value

A one-row data frame with diagnostics and metadata.

`nmar` *Not Missing at Random (NMAR) Estimation*

Description

High-level interface for NMAR estimation.

`nmar()` validates basic inputs and dispatches to an engine (for example `el_engine`). The engine controls the estimation method and interprets `formula`; see the engine documentation for model-specific requirements.

Usage

```
nmar(formula, data, engine, trace_level = 0)
```

Arguments

formula	A two-sided formula. Many engines support a partitioned right-hand side via |, for example y_miss ~ block1_vars | block2_vars. The meaning of these blocks is engine-specific (see the engine documentation). In the common "missing values indicate nonresponse" workflow, the left-hand side is the outcome with NA values for nonrespondents.
data	A data.frame or a survey.design containing the variables referenced by formula.
engine	An NMAR engine configuration object, typically created by el_engine, exptilt_engine, or exptilt_nonparam_engine. This object defines the estimation method and tuning parameters and must inherit from class "nmar_engine".
trace_level	Integer 0-3; controls verbosity during estimation (default 0): • 0: no output (silent mode); • 1: major steps only (initialization, convergence, final results); • 2: iteration summaries and key diagnostics; • 3: full diagnostic output.

Value

An object of class "nmar_result" with an engine-specific subclass (for example "nmar_result_el"). Use summary(), se, confint(), weights(), coef(), fitted(), and generics::tidy() / generics::glance() to access estimates, standard errors, weights, and diagnostics.

See Also

el_engine, exptilt_engine, exptilt_nonparam_engine, summary.nmar_result, weights.nmar_result

Examples

```
set.seed(1)
n <- 200
x1 <- rnorm(n)
z1 <- rnorm(n)
y_true <- 0.5 + 0.3 * x1 + 0.2 * z1 + rnorm(n, sd = 0.3)
resp <- rbinom(n, 1, plogis(2 + 0.1 * y_true + 0.1 * z1))
if (all(resp == 1)) resp[sample.int(n, 1)] <- 0L
y_obs <- ifelse(resp == 1, y_true, NA_real_)

# Empirical likelihood engine
df_el <- data.frame(Y_miss = y_obs, X = x1, Z = z1)
eng_el <- el_engine(variance_method = "none")
fit_el <- nmar(Y_miss ~ X | Z, data = df_el, engine = eng_el)
summary(fit_el)


# Exponential tilting engine (illustrative)
dat_et <- data.frame(y = y_obs, x2 = z1, x1 = x1)
eng_et <- exptilt_engine(
  y_dens = "normal",
  family = "logit",
```

```
  variance_method = "none"
)
fit_et <- nmar(y ~ x2 | x1, data = dat_et, engine = eng_et)
summary(fit_et)

# Survey design example (same outcome, random weights)
if (requireNamespace("survey", quietly = TRUE)) {
  w <- runif(n, 0.5, 2)
  des <- survey::svydesign(ids = ~1, weights = ~w,
                           data = data.frame(Y_miss = y_obs, X = x1, Z = z1))
  eng_svy <- el_engine(variance_method = "none")
  fit_svy <- nmar(Y_miss ~ X | Z, data = des, engine = eng_svy)
  summary(fit_svy)
}

# Bootstrap variance usage
if (requireNamespace("future.apply", quietly = TRUE)) {
  set.seed(2)
  eng_boot <- el_engine(
    variance_method = "bootstrap",
    bootstrap_reps = 20
  )
  fit_boot <- nmar(Y_miss ~ X | Z, data = df_el, engine = eng_boot)
  se(fit_boot)
}
```

---

nmar_engine_helpers *S3 helpers for NMAR engine objects*

Description

Lightweight, user-facing methods for engine configuration objects (class 'nmar_engine'). These improve discoverability and provide a consistent print surface across engines while keeping the objects as simple lists internally.

Design

- 'engine_name()' returns a canonical identifier used across the package (e.g., in 'nmar_result$meta$engine_name'). - 'print.nmar_engine()' provides a concise, readable summary of the engine configuration; engine-specific classes reuse the parent method unless they need to override it. - 'engine_config()' returns the underlying configuration as a named list for programmatic inspection.

`polish_households` *Polish Household Budget Data with Simulated Nonignorable Nonresponse*

Description

This dataset is derived from the ‘h05‘ dataset (Polish household budgets for 2005) found in the ‘RClas‘ package. The original data was cleaned to remove all rows with missing values.

Usage

```
polish_households
```

Format

A data frame with 19,330 rows and 17 columns. The key variables are:

class TODO

voi TODO

bio TODO

type TODO

d345 TODO

d347 TODO

d348 TODO

d36 TODO

d38 TODO

d61 TODO

noper TODO

income TODO

expenditure TODO

y_exp Numeric. The **true** scaled expenditure (‘expenditure / mean(expenditure)‘). This is the complete study variable without missingness.

resp TODO

R Integer. The simulated response indicator (1=responded, 0=nonresponse).

y_exp_miss Numeric. The **observed** scaled expenditure, containing 7,778 ‘NA‘ values where ‘R = 0‘. This is the variable to be used as the NMAR-affected outcome.

Details

To create a realistic test case for nonignorable nonresponse (NMAR), a nonresponse mechanism was simulated and applied to the scaled expenditure variable (‘y_exp‘).

The key simulation steps were: 1. ‘y_exp‘ (true study variable) was created by scaling total expenditure. 2. A true response probability (‘resp‘) was created using the logistic model: ‘plogis(1 - 0.6 * y_exp)‘. 3. A response indicator (‘R‘) was simulated based on this probability. 4. The final variable ‘y_exp_miss‘ was generated by setting ‘y_exp‘ to ‘NA‘ wherever ‘R‘ was 0.

The response is **nonignorable** because the probability of missingness depends directly on the value of the expenditure variable itself.

Source

TODO

See Also

‘riddles_case1‘, ‘riddles_case2‘, ‘riddles_case3‘, ‘riddles_case4‘

---

print.nmar_engine *Print method for NMAR engines*

Description

Provides a compact, human-friendly summary for ‘nmar_engine‘ objects. Child classes inherit this method; they can override it if they need a different presentation.

Usage

```
## S3 method for class 'nmar_engine'
print(x, ...)
```

Arguments

x	An engine object inheriting from ‘nmar_engine‘.
...	Unused.

Value

‘x‘, invisibly.

print.nmar_result *Print method for nmar_result*

Description

Print method for nmar_result

Usage

```
## S3 method for class 'nmar_result'
print(x, ...)
```

Arguments

x	nmar_result object
...	Additional parameters

Value

'x', invisibly.

print.nmar_result_el *Print method for EL results*

Description

Compact print for objects of class `nmar_result_el`.

Usage

```
## S3 method for class 'nmar_result_el'
print(x, ...)
```

Arguments

x	An object of class `nmar_result_el`.
...	Ignored.

Value

x, invisibly.

print.nmar_result_exptilt

Print method for Exponential Tilting results (engine-specific)

Description

This print method is tailored for ‘nmar_result_exptilt‘ objects and shows a concise, human-friendly summary of the estimation result together with exptilt-specific diagnostics (loss, iterations) and a compact view of the response coefficients stored in the fitted model.

Usage

```
## S3 method for class 'nmar_result_exptilt'
print(x, ...)
```

Arguments

x	An object of class ‘nmar_result_exptilt‘.
...	Ignored.

Value

‘x‘, invisibly.

print.summary_nmar_result

Print method for summary.nmar_result

Description

Print method for summary.nmar_result

Usage

```
## S3 method for class 'summary_nmar_result'
print(x, ...)
```

Arguments

x	summary_nmar_result object
...	Additional parameters

Value

‘x‘, invisibly.

`riddles_case1` *Riddles Simulation, Case 1: Linear Mean*

Description

A simulated dataset of 500 observations based on Simulation Study I (Model 1, Case 1) of Riddles, Kim, and Im (2016). The data features a nonignorable nonresponse (NMAR) mechanism where the response probability depends on the study variable ‘y‘.

Usage

```
riddles_case1
```

Format

A data frame with 500 rows and 4 variables:

x Numeric. The auxiliary variable, x ~ Normal(0, 0.5).

y Numeric. The study variable with nonignorable nonresponse. ‘y‘ contains ‘NA‘s for nonrespondents.

y_true Numeric. The complete, true value of ‘y‘ before missingness was introduced.

delta Integer. The response indicator (1 = responded, 0 = nonresponse).

Details

This dataset was generated using the following model parameters (n = 500):

Density for x: x ~ Normal(mean = 0, variance = 0.5)

Density for error: e ~ Normal(mean = 0, variance = 0.9)

True Model (Case 1): y_true = -1 + x + e

Response Model (NMAR): logit(pi) = 0.8 - 0.2 * y_true

Source

Riddles, M. K., Kim, J. K., & Im, J. (2016). A Propensity-Score-Adjustment Method for Nonignorable Nonresponse. Journal of Survey Statistics and Methodology, 4(1), 1-31.

`riddles_case2` *Riddles Simulation, Case 2: Exponential Mean*

Description

A simulated dataset of 500 observations based on Simulation Study I (Model 1, Case 2) of Riddles, Kim, and Im (2016). The data features a nonignorable nonresponse (NMAR) mechanism where the response probability depends on the study variable 'y'.

Usage

```
riddles_case2
```

Format

A data frame with 500 rows and 4 variables:

x Numeric. The auxiliary variable, x ~ Normal(0, 0.5).

y Numeric. The study variable with nonignorable nonresponse. 'y' contains 'NA's for nonrespondents.

y_true Numeric. The complete, true value of 'y' before missingness was introduced.

delta Integer. The response indicator (1 = responded, 0 = nonresponse).

Details

This dataset was generated using the following model parameters (n = 500):

Density for x: x ~ Normal(mean = 0, variance = 0.5)

Density for error: e ~ Normal(mean = 0, variance = 0.9)

True Model (Case 2): y_true = -2 + 0.5 * exp(x) + e

Response Model (NMAR): logit(pi) = 0.8 - 0.2 * y_true

Source

Riddles, M. K., Kim, J. K., & Im, J. (2016). A Propensity-Score-Adjustment Method for Nonignorable Nonresponse. Journal of Survey Statistics and Methodology, 4(1), 1-31.

`riddles_case3` *Riddles Simulation, Case 3: Sine Wave Mean*

Description

A simulated dataset of 500 observations based on Simulation Study I (Model 1, Case 3) of Riddles, Kim, and Im (2016). The data features a nonignorable nonresponse (NMAR) mechanism where the response probability depends on the study variable ‘y‘.

Usage

```
riddles_case3
```

Format

A data frame with 500 rows and 4 variables:

x Numeric. The auxiliary variable, x ~ Normal(0, 0.5).

y Numeric. The study variable with nonignorable nonresponse. ‘y‘ contains ‘NA‘s for nonrespondents.

y_true Numeric. The complete, true value of ‘y‘ before missingness was introduced.

delta Integer. The response indicator (1 = responded, 0 = nonresponse).

Details

This dataset was generated using the following model parameters (n = 500):

Density for x: x ~ Normal(mean = 0, variance = 0.5)

Density for error: e ~ Normal(mean = 0, variance = 0.9)

True Model (Case 3): y_true = -1 + sin(2 * x) + e

Response Model (NMAR): logit(pi) = 0.8 - 0.2 * y_true

Source

Riddles, M. K., Kim, J. K., & Im, J. (2016). A Propensity-Score-Adjustment Method for Nonignorable Nonresponse. Journal of Survey Statistics and Methodology, 4(1), 1-31.

`riddles_case4` *Riddles Simulation, Case 4: Cubic Mean*

Description

A simulated dataset of 500 observations based on Simulation Study I (Model 1, Case 4) of Riddles, Kim, and Im (2016). The data features a nonignorable nonresponse (NMAR) mechanism where the response probability depends on the study variable ‘y‘.

Usage

```
riddles_case4
```

Format

A data frame with 500 rows and 4 variables:

x Numeric. The auxiliary variable, x ~ Normal(0, 0.5).

y Numeric. The study variable with nonignorable nonresponse. ‘y‘ contains ‘NA‘s for nonrespondents.

y_true Numeric. The complete, true value of ‘y‘ before missingness was introduced.

delta Integer. The response indicator (1 = responded, 0 = nonresponse).

Details

This dataset was generated using the following model parameters (n = 500):

Density for x: x ~ Normal(mean = 0, variance = 0.5)

Density for error: e ~ Normal(mean = 0, variance = 0.9)

True Model (Case 4): y_true = -1 + 0.4 * x^3 + e

Response Model (NMAR): logit(pi) = 0.8 - 0.2 * y_true

Source

Riddles, M. K., Kim, J. K., & Im, J. (2016). A Propensity-Score-Adjustment Method for Nonignorable Nonresponse. Journal of Survey Statistics and Methodology, 4(1), 1-31.

se *Extract standard error for NMAR results*

Description

Returns the standard error of the primary mean estimate.

Usage

```
se(object, ...)
```

Arguments

object	An ‘nmar_result‘ or subclass.
...	Ignored.

Value

Numeric scalar.

summary.nmar_result *Summary method for nmar_result*

Description

Summary method for nmar_result

Usage

```
## S3 method for class 'nmar_result'
summary(object, conf.level = 0.95, ...)
```

Arguments

object	nmar_result object
conf.level	Confidence level for intervals.
...	Additional parameters

Value

An object of class ‘summary_nmar_result‘.

summary.nmar_result_el *Summary method for EL results*

Description

Summarize estimation, standard error and missingness-model coefficients.

Usage

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

Arguments

object	An object of class nmar_result_el.
...	Ignored.

Value

An object of class summary_nmar_result_el.

summary.nmar_result_exptilt *Summary method for Exponential Tilting results (engine-specific)*

Description

Summarize estimation, standard error and model coefficients.

Usage

```
## S3 method for class 'nmar_result_exptilt'
summary(object, conf.level = 0.95, ...)
```

Arguments

object	An object of class ‘nmar_result_exptilt‘.
conf.level	Confidence level for confidence interval (default 0.95).
...	Ignored.

Value

An object of class ‘summary_nmar_result_exptilt‘.

tidy.nmar_result *Tidy summary for NMAR results*

Description

Return a data frame with the primary estimate and (if available) missingness-model coefficients.

Usage

```
## S3 method for class 'nmar_result'
tidy(x, conf.level = 0.95, ...)
```

Arguments

x	An object of class ‘nmar_result‘.
conf.level	Confidence level for the primary estimate.
...	Ignored.

Value

A data frame with one row for the primary estimate and, when available, additional rows for the response-model coefficients.

vcov.nmar_result *Variance-covariance for base NMAR results*

Description

Variance-covariance for base NMAR results

Usage

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

Arguments

object	An object of class ‘nmar_result‘.
...	Ignored.

Value

A 1x1 numeric matrix (the variance of the primary estimate).

`voting` *Aggregated Exit Poll Data for Gangdong-Gap (2012)*

Description

This dataset contains the aggregated exit poll results for the Gangdong-Gap district in Seoul from the 2012 nineteenth South Korean legislative election. The data is transcribed directly from Table 9 of Riddles, Kim, and Im (2016).

Usage

```
voting
```

Format

A data frame with 8 rows and 7 variables:

Gender Factor. The gender of the voter ("Male", "Female").

Age_group Character. The age group of the voter.

Voted_A Numeric. Count of respondents voting for Party A.

Voted_B Numeric. Count of respondents voting for Party B.

Other Numeric. Count of respondents voting for another party.

Refusal Numeric. Count of sampled individuals who refused to respond (this is the nonresponse count).

Total Numeric. Total individuals sampled in the group (Responders + Refusals).

Details

In the paper's application, 'Gender' is used as the nonresponse instrumental variable and 'Age_group' is the primary auxiliary variable .

Source

Riddles, M. K., Kim, J. K., & Im, J. (2016). A Propensity-Score-Adjustment Method for Nonignorable Nonresponse. *Journal of Survey Statistics and Methodology*, 4(1), 1–31. (Data from Table 9, p. 20).

`weights.nmar_result` *Extract weights from an ‘nmar_result‘*

Description

Return analysis weights stored in an ‘nmar_result‘ as either probability-scale (summing to 1) or population-scale (summing to ‘sample$n_total‘). The function normalizes stored masses and attaches informative attributes.

Usage

```
## S3 method for class 'nmar_result'
weights(object, scale = c("probability", "population"), ...)
```

Arguments

`object`	An ‘nmar_result‘ object.
`scale`	One of ‘"probability"‘ (default) or ‘"population"‘.
`...`	Additional arguments (ignored).

Value

Numeric vector of weights with length equal to the number of respondents.

Index