Package: mimar (via r-universe)

June 9, 2026

Title Compact Multiple Imputation, Assessment, and Reporting

Version 0.8.0

Author Imad EL BADISY [aut, cre]

Maintainer Imad EL BADISY <elbadisyimad@gmail.com>

Description Provides compact tools for missing-data analysis, including artificial amputation, chained single and multiple imputation, statistical and machine-learning-based imputation methods, diagnostic evaluation, and post-imputation pooling.

License MIT + file LICENSE

URL https://github.com/ielbadisy/mimar

BugReports https://github.com/ielbadisy/mimar/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Imports BART, e1071, functionals, gbm, ggplot2, glmnet, missMDA, naivebayes, ranger, rpart, stats, tibble, xgboost

Suggests testthat (>= 3.0.0), knitr, rmarkdown, pkgload

Config/testthat/edition 3

VignetteBuilder knitr

Config/pak/sysreqs cmake make libicu-dev libuv1-dev libx11-dev zlib1g-dev

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

Date/Publication 2026-06-03 22:54:24 UTC

RemoteUrl https://github.com/ielbadisy/mimar

RemoteRef HEAD

RemoteSha 97a42db307ff46c005ff6e8351ef54fd46c40eb2

Contents

ampute.data.frame . . . 2
complete . . . 3
describe . . . 4
evaluate . . . 4
fit . . . 5
impute . . . 6
imputer . . . 8
imputer_registry . . . 9
plot.mimar_imputation . . . 10
pool . . . 11

Index **15**

`ampute.data.frame` *Artificially introduce missing data*

Description

`ampute()` creates benchmark data by adding missing values under MCAR, MAR, or MNAR mechanisms. The returned object keeps the original data, amputated data, and masks for original, added, and total missingness.

Usage

```
## S3 method for class 'data.frame'
ampute(
  x,
  prop = 0.1,
  mechanism = c("MCAR", "MAR", "MNAR"),
  target = NULL,
  by = NULL,
  direction = c("both", "left", "right"),
  seed = NULL,
  ...
)

ampute(x, ...)
```

Arguments

`x`	A data frame.
`prop`	Target marginal proportion of newly introduced missing values.
`mechanism`	Missingness mechanism: missing completely at random (`"MCAR"`), missing at random (`"MAR"`), or missing not at random (`"MNAR"`).
`target`	Character vector of variables eligible for amputation. Defaults to all variables.

by	Character vector of fully or partially observed variables used to drive MAR probabilities. Required for mechanism = "MAR".
direction	For MNAR numeric targets, whether missingness is more likely in both tails, the left tail, or the right tail.
seed	Optional random seed.
...	Passed to methods.

Details

For each target variable X_j, observed cells are removed by drawing

$$A_{ij} \sim \text{Bernoulli}(p_i)$$

and setting X_{ij} to missing when $A_{ij} = 1$. For MCAR, $p_i = \pi$, where π is prop. For MAR, probabilities are based on observed by variables through a standardized score s_i:

$$p_i = \text{logit}^{-1}(\alpha + s_i),$$

with α calibrated so that the average probability is prop. For MNAR, s_i is derived from the target variable itself; direction selects high values, low values, or both tails for numeric targets.

Value

A mimar_amputation object.

---

complete *Extract completed imputed data*

Description

complete() extracts completed data sets from imputation objects. For mimar_imputation objects, use complete(x) or complete(x, 1) for one completed data set, complete(x, "all") for all completed data sets, and complete(x, "long") for a stacked long data frame.

Usage

```
complete(x, ...)
```

Arguments

x	An object containing completed data.
...	Passed to methods.

Value

A data frame or list of data frames.

describe *Describe missing data and mimar objects*

Description

`describe()` summarizes missingness for data frames and returns compact summaries for `mimar` objects.

Usage

```
describe(x, ...)
```

Arguments

`x`	An object to describe.
`...`	Passed to methods.

Details

For a data matrix $X \in R^{n \times p}$, missingness is represented by the indicator

$$R_{ij} = I(X_{ij} \text{ is missing}),$$

where i indexes rows and j indexes variables. The variable-level missingness proportion reported by `describe()` is

$$\hat{\pi}_j = n^{-1} \sum_{i=1}^{n} R_{ij}.$$

Row summaries use $p^{-1} \sum_{j=1}^{p} R_{ij}$, and missingness patterns are the unique row vectors of R.

Value

A mimar S3 object.

evaluate *Evaluate imputation quality*

Description

`evaluate()` compares imputed values with known truth when an amputation object is available. Numeric targets are summarized with errors such as

$$e_{ij} = \tilde{X}_{ij} - X_{ij},$$

while categorical targets are summarized with agreement and balanced accuracy across classes. When no truth is available, `evaluate()` reports diagnostics that can be computed from the imputed data alone. Distribution, variability, and recovery summaries are computed across all completed data sets; per-imputation recovery metrics are kept in `recovery_by_imputation`.

Usage

```
evaluate(x, ...)
```

Arguments

x	A mimar_imputation or mimar_amputation object.
...	Passed to methods.

Value

A mimar_evaluation object.

---

fit *Fit an imputer learner*

Description

fit() trains a mimar_imputer on complete observed rows for one target variable. In the imputation loop, x is the predictor block X_{-j}^{obs} and y is the observed target vector X_j^{obs}.

Usage

```
fit(object, ...)

## S3 method for class 'mimar_imputer'
fit(
  object,
  x,
  y,
  target = y,
  variable = "target",
  donors = 5,
  seed = NULL,
  ...
)
```

Arguments

object	Object to fit.
...	Passed to methods.
x	Predictor data frame containing observed rows for the current target variable.
y	Observed target vector for the current variable.
target	Original target vector, used to validate type support and restore imputed values to the correct storage type.
variable	Variable name used in diagnostics and error messages.
donors	Number of predictive mean matching donors for "pmm".
seed	Optional random seed.

Details

The fitted object stores the original imputer descriptor and the model needed by `predict()`. Native imputers are implemented directly in `mimar`; wrapped imputers call their original learner packages directly:

- `"norm"` fits a linear model and draws $\tilde{y} = \hat{y} + \epsilon$, with $\epsilon \sim N(0, \hat{\sigma}^2)$.
- `"pmm"` fits the same linear model but imputes by predictive mean matching: among observed donors with fitted values closest to $\hat{y}_{mis}$, one donor value is sampled.
- `"logreg"` fits a binomial GLM and draws classes from fitted Bernoulli probabilities.
- `"polyreg"` fits one-vs-rest binomial GLMs and samples classes from normalized class probabilities.

Value

A fitted object.

Methods (by class)

- `fit(mimar_imputer)`: Fit a `mimar_imputer`.

`impute` *Impute missing data*

Description

`impute()` performs single or multiple imputation through a chained update procedure owned by `mimar`. The default `imputer = "pmm"` uses predictive mean matching. A named imputer such as `"naive"`, `"rf"`, `"xgboost"`, `"knn"`, or `"glmnet"` can also be supplied to use that learner for every incomplete variable it supports. The returned object keeps completed datasets first; use `complete()` to extract one completed dataset, all completed datasets, or a stacked long data frame.

Usage

```
impute(
  x,
  m = 5,
  imputer = "pmm",
  maxit = 5,
  seed = NULL,
  donors = 5,
  ncore = 1,
  verbose = FALSE,
  ...
)

## S3 method for class 'data.frame'
```

```
impute(
  x,
  m = 5,
  imputer = "pmm",
  maxit = 5,
  seed = NULL,
  donors = 5,
  ncore = 1,
  verbose = FALSE,
  ...
)

## S3 method for class 'mimar_amputation'
impute(
  x,
  m = 5,
  imputer = "pmm",
  maxit = 5,
  seed = NULL,
  donors = 5,
  ncore = 1,
  verbose = FALSE,
  ...
)
```

Arguments

x	A data frame or `mimar_amputation` object.
m	Number of completed data sets to generate.
imputer	Imputer name or a `mimar_imputer` specification. Any name returned by `imputer_registry()` can be supplied to use that imputer for all incomplete variables it supports.
maxit	Number of chained-equation iterations.
seed	Optional random seed.
donors	Number of donor candidates used by donor-based imputers.
ncore	Number of CPU cores used to run completed datasets in parallel. The default, `1`, runs sequentially. Values greater than one are passed to `functionals::fmap()` as `ncores`.
verbose	Logical; if `TRUE`, print an informative progress log for the chained-imputation workflow. The default, `FALSE`, runs silently.
...	Passed to methods.

Details

Hyperparameters for learner-backed imputers can be supplied through the `imputer()` specification or directly through `...` when calling `impute()`. For donor-based imputers, `donors` controls the donor pool used by `"pmm"`, `"spmm"`, `"knn"`, and `"hotdeck"`.

Let X_j be an incomplete variable and X_{-j} all remaining variables. At each iteration, `mimar` fits an imputer learner on observed rows

$$\hat{f}_j : X_{-j}^{obs} \to X_j^{obs}$$

and predicts missing cells X_j^{mis} from X_{-j}^{mis}. This is repeated across incomplete variables for `maxit` iterations and across m independent completed data sets. The algorithm is intentionally learner-agnostic: each imputer is constructed with `imputer()`, trained with `fit()`, and used through `predict()`.

Each requested imputer is applied to all incomplete variables it supports. Use `imputer_registry()` to inspect target-type compatibility. Learner-backed methods are supervised stochastic update rules inside the chained workflow; inspect diagnostics and downstream sensitivity rather than treating any single learner as a guarantee of proper uncertainty quantification.

Value

A `mimar_imputation` object.

Methods (by class)

- `impute(data.frame)`: Impute a data frame.
- `impute(mimar_amputation)`: Impute a `mimar_amputation` object and retain truth masks for later evaluation.

`imputer` *Build an imputer learner*

Description

`imputer()` constructs a standalone learner descriptor used by the chained imputation engine. All imputers expose the same standard lifecycle:

Usage

```
imputer(method, ...)

## Default S3 method:
imputer(method, spec = NULL, ...)
```

Arguments

`method`	Imputer method name.
`...`	Hyperparameters retained for later use by `impute()` or `fit()`.
`spec`	Optional learner specification retained for future extensions.

Details

1. construct with `imputer(method)`;
2. fit on observed rows with `fit(object, x, y)`;
3. impute new rows with `predict(fitted, newdata)`.

Native learners implemented in `mimar` include `"mean"`, `"median"`, `"mode"`, `"naive"`, `"norm"`, `"pmm"`, `"spmm"`, `"logreg"`, `"polyreg"`, `"knn"`, and `"hotdeck"`. Learner-backed imputers such as `"rf"`, `"xgboost"`, `"svm"`, `"bart"`, `"nbayes"`, `"rpart"`, `"glmnet"`, `"gbm"`, and `"famd"` are called directly through their original packages installed with `mimar`. Additional arguments supplied to `imputer()` are retained as hyperparameters and used by `impute()` and `fit()`. The `"superlearner"` imputer, also available as `"sl"`, cross-validates a candidate imputer library on observed cells and combines candidates using non-negative loss-based weights.

Compatibility with target types is explicit. If an imputer does not support a numeric, binary, or multiclass target, `mimar` stops with an error rather than silently falling back to another method.

Value

A `mimar_imputer` object.

Methods (by class)

- `imputer(default)`: Construct a `mimar_imputer`.

`imputer_registry` *List available mimar imputers*

Description

`imputer_registry()` returns the imputer names accepted by `impute()` and metadata describing target-type support and backend packages. Native `mimar` methods display `package = "internal"`. The result is returned as a tibble.

Usage

```
imputer_registry()
```

Value

A tibble.

`plot.mimar_imputation` *Diagnostic plots for mimar imputations*

Description

`plot.mimar_imputation()` draws imputation diagnostics. By default it shows imputed cell counts. Other plot types show a cell-status map, observed versus imputed distributions, boxplots across imputations, bivariate diagnostics, categorical proportions, convergence traces, variable-level imputation methods, or between-imputation variability.

Usage

```
## S3 method for class 'mimar_imputation'
plot(
  x,
  type = c("imputed", "missing", "density", "strip", "boxplot", "xy", "proportion",
    "trace", "methods", "variability"),
  variable = NULL,
  formula = NULL,
  statistic = c("mean", "sd"),
  ...
)
```

Arguments

`x`	A `mimar_imputation` object.
`type`	Plot type: `"imputed"`, `"missing"`, `"density"`, `"strip"`, `"boxplot"`, `"xy"`, `"proportion"`, `"trace"`, `"methods"`, or `"variability"`.
`variable`	Optional variable name or names used by distribution plots.
`formula`	Optional formula for bivariate and stratified diagnostics. Use forms such as `height ~ weight`, `height ~ weight | gender`, or `group ~ sex`.
`statistic`	Trace statistic, either `"mean"` or `"sd"`.
`...`	Unused.

Value

A `ggplot` object.

pool *Pool post-fit quantities across imputations*

Description

pool() combines post-fit quantities estimated separately on each completed data set. The object being pooled is a quantity: a scalar, vector, matrix, array, model coefficient, survival probability, metric, or other estimate of interest. A data frame is only a convenient tabular adapter for tidy scalar estimates; it is not the statistical target being pooled.

Usage

```
pool(x, ...)

## S3 method for class 'numeric'
pool(
  x,
  variance = NULL,
  std.error = NULL,
  covariance = NULL,
  rule = NULL,
  transform = NULL,
  inverse = NULL,
  conf.level = 0.95,
  name = "quantity",
  ...
)

## S3 method for class 'list'
pool(
  x,
  variance = NULL,
  std.error = NULL,
  covariance = NULL,
  rule = NULL,
  transform = NULL,
  inverse = NULL,
  conf.level = 0.95,
  ...
)

## S3 method for class 'matrix'
pool(
  x,
  variance = NULL,
  std.error = NULL,
  covariance = NULL,
```

```
  rule = NULL,
  transform = NULL,
  inverse = NULL,
  conf.level = 0.95,
  ...
)

## S3 method for class 'data.frame'
pool(
  x,
  variance = NULL,
  std.error = NULL,
  covariance = NULL,
  rule = NULL,
  transform = NULL,
  inverse = NULL,
  conf.level = 0.95,
  ...
)
```

Arguments

x	Quantity estimates across imputations. Use a numeric vector for one scalar quantity, a list for scalar/vector/matrix/array quantities, a matrix with imputations in rows and quantities in columns, or a data frame as a tabular adapter for tidy scalar estimates.
...	Passed to methods.
variance	Complete-data variance for the quantity in each imputation. For vector quantities this may also be a list of elementwise variance vectors or matrices matching x.
std.error	Complete-data standard error for the quantity in each imputation. Ignored when variance is supplied.
covariance	For a list of numeric vectors, optional list of covariance matrices. When supplied, vector pooling uses Rubin's multivariate matrix form and returns the pooled covariance matrix.
rule	Pooling rule. "rubin" applies Rubin's rules and requires variance, std.error, or covariance. "robust" reports median, IQR, and range across imputations. "mean" reports the mean and between-imputation standard error. Defaults to "rubin" when variance is available and "robust" otherwise.
transform	Optional function applied before pooling, for example log, qlogis, or function(p) log(-log(p)).
inverse	Optional inverse transformation applied to pooled estimates and intervals.
conf.level	Confidence level for interval estimates.
name	Name of a scalar quantity.

Details

For a scalar quantity with estimates $Q_1, \ldots, Q_m$ and complete-data variances $U_1, \ldots, U_m$, Rubin rules are

$$\bar{Q} = m^{-1} \sum_{k=1}^{m} Q_k,$$

$$\bar{U} = m^{-1} \sum_{k=1}^{m} U_k,$$

$$B = (m-1)^{-1} \sum_{k=1}^{m} (Q_k - \bar{Q})^2,$$

and total variance is

$$T = \bar{U} + (1 + m^{-1})B.$$

The reported standard error is $\sqrt{T}$. Confidence intervals use a t reference distribution with

$$\nu = (m-1)\left(1 + r^{-1}\right)^2, \quad r = \frac{(1 + m^{-1})B}{\bar{U}}.$$

For a vector quantity, pass `x` as a list of numeric vectors and `covariance` as a list of complete-data covariance matrices. Rubin's matrix form is then used: $\bar{Q}$ is a vector and $T = \bar{U} + (1 + m^{-1})B$ is the pooled covariance matrix. For matrices or arrays, pass a list of same-shaped quantities. Unless a joint covariance structure is supplied through a vector input, these are pooled element by element, which is appropriate for grids of scalar estimands such as survival probabilities at several times and covariate profiles.

Some metrics do not have reliable complete-data variance estimates or do not satisfy approximate normality. Following Marshall et al. (2009), `pool()` reports robust summaries by default when no variance is supplied: median, interquartile range, and range across imputations. Use `rule = "mean"` to request a mean and between-imputation standard error for such metrics.

A list is the preferred input for post-fit quantities. Use a list of length `m`, one element per imputation. Each element can be a scalar, vector, matrix, or array. When `covariance` is supplied for a list of vectors, Rubin's multivariate matrix rule is used. Otherwise list elements are pooled element by element.

Data frames are accepted as a convenience adapter, but the pooled object is not the data frame itself. Rows must encode post-fit scalar quantities: `term`, `estimate`, `std.error`, and `imputation` for Rubin pooling, or `metric`, `value`, and `imputation` for metric summaries.

Value

A `mimar_pool` object.

Methods (by class)

- `pool(numeric)`: Pool a scalar quantity observed across imputations.
- `pool(list)`: Pool a list of scalar, vector, matrix, or array quantities.
- `pool(matrix)`: Pool a matrix whose rows are imputations and columns are scalar quantities.
- `pool(data.frame)`: Tabular adapter for tidy scalar estimates or metrics.

References

Marshall A, Altman DG, Holder RL, Royston P. Combining estimates of interest in prognostic modelling studies after multiple imputation: current practice and guidelines. BMC Medical Research Methodology. 2009;9:57.

Examples

```
pool(c(0.10, 0.11, 0.09), std.error = c(0.04, 0.05, 0.04), name = "age")

betas <- list(c(age = 0.10, bmi = 0.30),
              c(age = 0.11, bmi = 0.32),
              c(age = 0.09, bmi = 0.29))
covs <- list(diag(c(0.04, 0.08)^2),
             diag(c(0.05, 0.09)^2),
             diag(c(0.04, 0.08)^2))
pool(betas, covariance = covs)
```

Index

ampute *(*ampute.data.frame*)*, 2
ampute.data.frame, 2

complete, 3

describe, 4

evaluate, 4

fit, 5

impute, 6
imputer, 8
imputer_registry, 9

plot.mimar_imputation, 10
pool, 11