Package 'missForest'

May 8, 2026

Type Package

Title Nonparametric Missing Value Imputation using Random Forest

Version 1.6.1

Date 2025-10-22

Maintainer Daniel J. Stekhoven <stekhoven@nexus.ethz.ch>

Imports randomForest, ranger, foreach, iterators, itertools, doRNG, stats, Rdpack

Suggests doParallel, knitr, rmarkdown

VignetteBuilder knitr

Description The function 'missForest' in this package is used to impute missing values particularly in the case of mixed-type data. It uses a random forest (via 'ranger' or 'randomForest') trained on the observed values of a data matrix to predict the missing values. It can be used to impute continuous and/or categorical data including complex interactions and non-linear relations. It yields an out-of-bag (OOB) imputation error estimate without the need of a test set or elaborate cross-validation. It can be run in parallel to save computation time.

License GPL (>= 2)

URL https://www.r-project.org, https://github.com/stekhoven/missForest

BugReports https://github.com/stekhoven/missForest/issues

RdMacros Rdpack

Encoding UTF-8

NeedsCompilation no

Author Daniel J. Stekhoven [aut, cre]

Repository CRAN

Date/Publication 2025-10-26 12:30:02 UTC

Contents

`missForest-package` *Nonparametric Missing Value Imputation using Random Forest (ranger by default)*

Description

The **missForest** package provides nonparametric missing-value imputation for mixed-type data (continuous and categorical). It models each variable with missingness using random forests that learn complex interactions and nonlinear relations and returns out-of-bag (OOB) error estimates. The default backend is **ranger** for speed and scalability, with an optional legacy **randomForest** backend for backward compatibility. Parallelization is supported either across variables (via **foreach/doRNG**) or within forests (via **ranger** threads).

Details

Package:	missForest
Type:	Package
Version:	1.6
Date:	2025-10-13
License:	GPL (>= 2)

The main function is `missForest`, which iteratively imputes missing entries by fitting per-variable random forests to the currently imputed data matrix. The implementation now defaults to a **ranger**-based backend while preserving the original **randomForest**-based behavior via the `backend` argument. See `missForest` for arguments, details on stopping criteria, OOB error reporting (NRMSE for numeric variables and PFC for factors), and parallel options.

Author(s)

Daniel J. Stekhoven [aut, cre]

References

Stekhoven DJ, Bühlmann P (2012). "MissForest — nonparametric missing value imputation for mixed-type data." *Bioinformatics*, **28**(1), 112–118. doi:10.1093/bioinformatics/btr597.

See Also

missForest, mixError, prodNA

---

missForest *Nonparametric Missing Value Imputation using Random Forests (ranger or randomForest)*

Description

missForest imputes missing values for mixed-type data (numeric and categorical). It models complex interactions and nonlinear relations and returns an out-of-bag (OOB) imputation error estimate. It supports parallel execution and offers two backends: **ranger** (default) and **randomForest** (legacy/compatibility).

Usage

```
missForest(xmis, maxiter = 10, ntree = 100, variablewise = FALSE,
           decreasing = FALSE, verbose = FALSE,
           mtry = floor(sqrt(ncol(xmis))), replace = TRUE,
           classwt = NULL, cutoff = NULL, strata = NULL,
           sampsize = NULL, nodesize = NULL, maxnodes = NULL,
           xtrue = NA, parallelize = c("no", "variables", "forests"),
           num.threads = NULL, backend = c("ranger", "randomForest"))
```

Arguments

xmis	A data frame or matrix with missing values. Columns are variables, rows are observations. All columns must be numeric or factor (character columns should be converted to factors beforehand).
maxiter	Maximum number of iterations unless the stopping criterion is met earlier.
ntree	Number of trees to grow in each per-variable forest.
variablewise	Logical. If TRUE, return an OOB error per variable; otherwise report one error for numeric variables (NRMSE) and one for factors (PFC).
decreasing	Logical. If FALSE, variables are processed in increasing order of missingness.
verbose	Logical. If TRUE, print iteration-wise diagnostics (estimated error, runtime, and—if xtrue is given—the true error).
mtry	Number of candidate variables at each split. Passed to the backend (**randomForest** or **ranger**). Default is $\sqrt{p}$.
replace	Logical. If TRUE, bootstrap sampling (with replacement) is used; otherwise subsampling (without replacement).
classwt	List of class priors for the categorical variables. Same list semantics as in **randomForest**: one element per variable (set NULL for numeric variables). With backend "ranger", this maps to class.weights.

`cutoff` List of per-class cutoff vectors for each categorical variable. As in **randomForest**, one element per factor variable. With backend `"ranger"`, cutoffs are emulated by fitting a probability forest and thresholding predicted class probabilities post-hoc.

`strata` List of (factor) variables used for stratified sampling (legacy **randomForest** semantics). Ignored by **ranger**.

`sampsize` List of sample sizes per variable (legacy **randomForest** semantics). With backend `"ranger"`, these are converted to `sample.fraction` (overall or per-class fractions, as appropriate).

`nodesize` Minimum node size. A numeric vector of length 2: *first* entry for **numeric** variables, *second* for **factor** variables. **Default:** `c(5, 1)`. With backend `"ranger"`, this maps to `min.bucket` (no exact 1:1 mapping to **randomForest**'s terminal-node semantics).

`maxnodes` Maximum number of terminal nodes per tree. Used with backend `"randomForest"`. With `"ranger"`, this argument is ignored (consider `max.depth` at the **ranger** level if needed).

`xtrue` Optional complete data matrix for benchmarking. If provided, the iteration log includes the true imputation error, and the return value includes it as `$error`.

`parallelize` Should `missForest` run in parallel? One of `"no"`, `"variables"`, or `"forests"`.

`"variables"` Forests for different variables are built in parallel using a registered **foreach** backend.

`"forests"` Within a variable, the forest is built using the backend's threading (for `"ranger"`) or via **foreach** sub-forests (for `"randomForest"`).

Which choice is faster depends on data shape and backend.

`num.threads` Integer (or `NULL`). Number of threads for **ranger**. If `parallelize = "variables"`, per-variable **ranger** calls use `num.threads = 1` internally to avoid nested oversubscription. Otherwise, if `NULL`, **ranger**'s default is used. Ignored by `"randomForest"`.

`backend` Character. `"ranger"` (default) uses **ranger** for forest fitting; `"randomForest"` retains legacy behavior for compatibility.

Details

Algorithm. The method iteratively imputes each variable with missing values by fitting a random forest on the observed part of that variable and the current imputations of all other variables. After each iteration, the difference between the current and previous imputed matrices is computed separately for numeric and factor columns. The stopping rule is met once both differences have increased at least once (or only the present type increases if there is only one type). In that case, the *previous* imputation (before the increase) is returned. Otherwise, the process stops at `maxiter`.

Backends. With `backend = "ranger"`, arguments are mapped as:

- `ntree` -> `num.trees`
- `nodesize` (numeric/factor) -> `min.bucket` for regression/classification, respectively (defaults used here are `c(5, 1)`).
- `sampsize` (counts) -> `sample.fraction` (overall or per-class fractions).
- `classwt` -> `class.weights`.

- `cutoff`: emulated via probability forests and post-thresholding.
- `maxnodes`: no direct equivalent in **ranger** (ignored).

The reported OOB error uses **ranger**'s `$prediction.error` (MSE for numeric, error rate for factors), except when `cutoff` is used: in that case, the misclassification rate is computed by applying the cutoffs to OOB class probabilities.

Parallelization. Two modes are available via `parallelize`:

- `"variables"`: different variables are imputed in parallel using **foreach**; per-variable **ranger** calls use `num.threads = 1`.
- `"forests"`: a single variable's forest is built using **ranger** multithreading (controlled by `num.threads`) or, for `"randomForest"`, by combining sub-forests via **foreach**.

Make sure you have registered a parallel backend if you choose a parallel mode.

See the vignette for further examples and discussion.

Value

`ximp`	Imputed data matrix (same classes as `xmis`).
`OOBerror`	Estimated OOB imputation error. For numeric variables, the normalized root mean squared error (NRMSE); for factors, the proportion falsely classified (PFC). If `variablewise = TRUE`, a vector of length p with per-variable errors is returned (labeled `"MSE"` for numeric and `"PFC"` for factors).
`error`	True imputation error (NRMSE/PFC), present only if `xtrue` was given.

Author(s)

Daniel J. Stekhoven [aut, cre]

References

Stekhoven DJ, Bühlmann P (2012). "MissForest — nonparametric missing value imputation for mixed-type data." *Bioinformatics*, **28**(1), 112–118. doi:10.1093/bioinformatics/btr597.

See Also

`mixError`, `prodNA`, `randomForest`, `ranger`

Examples

```
## Mixed-type imputation on iris:
data(iris)
set.seed(81)
iris.mis <- prodNA(iris, noNA = 0.2)

## Default: ranger backend
imp_rg <- missForest(iris.mis, xtrue = iris, verbose = TRUE)
imp_rg$OOBerror
imp_rg$error  # requires xtrue
```

```
## Legacy behavior: randomForest backend
imp_rf <- missForest(iris.mis, backend = "randomForest", verbose = TRUE)

## Parallel examples (register a backend first, e.g., doParallel):
## Not run:
# library(doParallel)
# registerDoParallel(2)
# imp_vars <- missForest(iris.mis, parallelize = "variables", verbose = TRUE)
# imp_fors <- missForest(iris.mis, parallelize = "forests", verbose = TRUE,
#                        num.threads = 2)  # used by ranger
## End(Not run)
```

`mixError` *Compute Imputation Error for Mixed-type Data*

Description

`mixError` computes imputation error for mixed-type data given the imputed matrix (`ximp`), the original matrix with missing values (`xmis`), and the complete ground truth (`xtrue`). It reports the normalized root mean squared error (NRMSE) for numeric variables and the proportion of falsely classified entries (PFC) for factor variables.

Usage

```
mixError(ximp, xmis, xtrue)
```

Arguments

`ximp`	Imputed data matrix (or data frame) with variables in columns and observations in rows. There must be no missing values.
`xmis`	Data matrix (or data frame) with missing values used to derive the missingness pattern.
`xtrue`	Complete data matrix (or data frame) containing the true values. There must be no missing values.

Value

A named vector with the imputation error(s):

- `NRMSE`: normalized root mean squared error computed over the numeric entries that were missing in `xmis`.
- `PFC`: proportion of falsely classified entries computed over the factor entries that were missing in `xmis`.

If only one type (numeric or factor) is present among the missing entries, only the corresponding error is returned.

Note

Columns are treated by their R classes: numeric metrics are computed for `numeric` columns and classification metrics for `factor` columns. Character columns should be converted to factors beforehand.

This function is used internally by `missForest` when a complete matrix `xtrue` is supplied.

Author(s)

Daniel J. Stekhoven [aut, cre]

References

Stekhoven DJ, Bühlmann P (2012). "MissForest — nonparametric missing value imputation for mixed-type data." *Bioinformatics*, **28**(1), 112–118. doi:10.1093/bioinformatics/btr597.

For the NRMSE notion in imputation benchmarking: Oba S, Sato M, Takemasa I, Monden M, Matsubara K, Ishii S (2003). "A Bayesian missing value estimation method for gene expression profile data." *Bioinformatics*, **19**(16), 2088–2096.

See Also

`missForest`, `nrmse`

Examples

```
## Mixed-type error computation on iris:
data(iris)

## Introduce missingness:
set.seed(81)
iris.mis <- prodNA(iris, noNA = 0.2)

## Impute:
iris.imp <- missForest(iris.mis)

## Compute the true imputation error:
err.imp <- mixError(iris.imp$ximp, iris.mis, iris)
err.imp
```

---

nrmse *Normalized root mean squared error*

Description

`nrmse` computes the normalized root mean squared error (NRMSE) for a given complete data matrix `xtrue`, an imputed matrix `ximp`, and the corresponding matrix with missing values `xmis`.

Usage

```
nrmse(ximp, xmis, xtrue)
```

Arguments

`ximp`	Imputed data matrix (or data frame) with variables in columns and observations in rows. Must be numeric and contain no missing values.
`xmis`	Data matrix (or data frame) with the original missing values. Its dimensions and column order must match `ximp` and `xtrue`.
`xtrue`	Complete data matrix (or data frame). Must be numeric and contain no missing values. Dimensions and column order must match `xmis`/`ximp`.

Details

The NRMSE is computed over the entries that were missing in `xmis` and are numeric in `xtrue` / `ximp`, using

$$\mathrm{NRMSE} = \sqrt{\frac{\mathrm{mean}\{(X_{\mathrm{true}} - X_{\mathrm{imp}})^2\}}{\mathrm{var}(X_{\mathrm{true}})}},$$

where mean and var are the empirical mean and variance computed over the continuous missing entries only. This measure is intended for continuous data; for categorical or mixed-type data, see `mixError`.

Value

A numeric scalar: the normalized root mean squared error.

Note

This function is used internally by `mixError`.

Author(s)

Daniel J. Stekhoven [aut, cre]

References

Oba S, Sato M, Takemasa I, Monden M, Matsubara K, Ishii S (2003). "A Bayesian missing value estimation method for gene expression profile data." *Bioinformatics*, **19**(16), 2088–2096.

See Also

`mixError`, `missForest`

Examples

```
## Simple numeric example
set.seed(1)
xtrue <- matrix(rnorm(100), ncol = 5)
xmis  <- xtrue
xmis[sample(length(xmis), 10)] <- NA
ximp  <- xmis
ximp[is.na(ximp)] <- rowMeans(ximp, na.rm = TRUE)[row(ximp)[is.na(ximp)]]

nrmse(ximp, xmis, xtrue)
```

---

prodNA *Introduce Missing Values Completely at Random (MCAR)*

Description

prodNA artificially introduces missing values by deleting entries completely at random (MCAR) up to a specified proportion.

Usage

```
prodNA(x, noNA = 0.1)
```

Arguments

x	A data frame or matrix to which missing values will be added. Column classes are preserved; factors receive NA entries.
noNA	Proportion of entries in x to set to NA. Must be a number in $[0, 1]$. The default is 0.1 (10% missingness).

Details

Missingness is introduced independently and uniformly over all cells, i.e., Missing Completely At Random (MCAR). No structure by row/column or variable type is imposed.

For reproducibility, call set.seed before prodNA.

Value

An object of the same base type as x (data frame or matrix) with approximately noNA proportion of its entries set to NA.

Author(s)

Daniel J. Stekhoven [aut, cre]

See Also

missForest, mixError

Examples

```
data(iris)

## Introduce 5% MCAR missingness into the iris data set:
set.seed(81)
iris.mis <- prodNA(iris, noNA = 0.05)
summary(iris.mis)

## Higher missingness:
set.seed(81)
iris.mis.20 <- prodNA(iris, noNA = 0.20)
mean(is.na(as.matrix(iris.mis.20)))
```

---

varClass *Extract Variable Types from a Data Frame*

Description

`varClass` returns the variable types of a data frame. It is used internally in several functions of the **missForest** package.

Usage

```
varClass(x)
```

Arguments

`x`	A data frame with variables in the columns.

Value

A character vector of length p, where p is the number of columns in `x`. Entries are `"numeric"` for continuous variables and `"factor"` for categorical variables.

Note

This function is used internally by `missForest` and `mixError`.

Author(s)

Daniel J. Stekhoven [aut, cre]

See Also

`missForest`, `mixError`, `nrmse`

Examples

```
data(iris)
varClass(iris)

## We have four continuous and one categorical variable.
```

Index