Package 'missRanger'

May 8, 2026

Title Fast Imputation of Missing Values

Version 2.6.1

Description Alternative implementation of the beautiful 'MissForest' algorithm used to impute mixed-type data sets by chaining random forests, introduced by Stekhoven, D.J. and Buehlmann, P. (2012) <doi:10.1093/bioinformatics/btr597>. Under the hood, it uses the lightning fast random forest package 'ranger'. Between the iterative model fitting, we offer the option of using predictive mean matching. This firstly avoids imputation with values not already present in the original data (like a value 0.3334 in 0-1 coded variable). Secondly, predictive mean matching tries to raise the variance in the resulting conditional distributions to a realistic level. This would allow, e.g., to do multiple imputation when repeating the call to missRanger(). Out-of-sample application is supported as well.

License GPL (>= 2)

Depends R (>= 3.5.0)

Encoding UTF-8

RoxygenNote 7.3.2

Imports FNN, ranger (>= 0.16.0), stats, utils

URL https://github.com/mayer79/missRanger, https://mayer79.github.io/missRanger/

BugReports https://github.com/mayer79/missRanger/issues

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Michael Mayer [aut, cre]

Maintainer Michael Mayer <mayermichael79@gmail.com>

Repository CRAN

Date/Publication 2024-12-07 09:20:02 UTC

Contents

`generateNA` *Adds Missing Values*

Description

Takes a vector, matrix or `data.frame` and replaces some values by `NA`.

Usage

```
generateNA(x, p = 0.1, seed = NULL)
```

Arguments

`x`	A vector, matrix or `data.frame`.
`p`	Proportion of missing values to add to `x`. In case `x` is a `data.frame`, `p` can also be a vector of probabilities per column or a named vector.
`seed`	An integer seed.

Value

`x` with missing values.

Examples

```
generateNA(1:10, p = 0.5)
head(generateNA(iris, p = 0.2))
```

imputeUnivariate *Univariate Imputation*

Description

Fills missing values of a vector, matrix or data frame by sampling with replacement from the non-missing values. For data frames, this sampling is done within column.

Usage

```
imputeUnivariate(x, v = NULL, seed = NULL)
```

Arguments

x	A vector, matrix or data frame.
v	A character vector of column names to impute (only relevant if x is a data frame). The default NULL imputes all columns.
seed	An integer seed.

Value

x with imputed values.

Examples

```
imputeUnivariate(c(NA, 0, 1, 0, 1))
head(imputeUnivariate(generateNA(iris)))
```

missRanger *Fast Imputation of Missing Values by Chained Random Forests*

Description

Uses the "ranger" package (Wright & Ziegler) to do fast missing value imputation by chained random forests, see Stekhoven & Buehlmann and Van Buuren & Groothuis-Oudshoorn. Between the iterative model fitting, it offers the option of predictive mean matching. This firstly avoids imputation with values not present in the original data (like a value 0.3334 in a 0-1 coded variable). Secondly, predictive mean matching tries to raise the variance in the resulting conditional distributions to a realistic level. This allows to do multiple imputation when repeating the call to missRanger().

Usage

```
missRanger(
  data,
  formula = . ~ .,
  pmm.k = 0L,
  num.trees = 500,
  mtry = NULL,
  min.node.size = NULL,
  min.bucket = NULL,
  max.depth = NULL,
  replace = TRUE,
  sample.fraction = if (replace) 1 else 0.632,
  case.weights = NULL,
  num.threads = NULL,
  save.memory = FALSE,
  maxiter = 10L,
  seed = NULL,
  verbose = 1,
  returnOOB = FALSE,
  data_only = !keep_forests,
  keep_forests = FALSE,
  ...
)
```

Arguments

`data`	A `data.frame` with missing values to impute.
`formula`	A two-sided formula specifying variables to be imputed (left hand side) and variables used to impute (right hand side). Defaults to `. ~ .`, i.e., use all variables to impute all variables. For instance, if all variables (with missings) should be imputed by all variables except variable "ID", use `. ~ . - ID`. Note that a "." is evaluated separately for each side of the formula. Further note that variables with missings must appear in the left hand side if they should be used on the right hand side.
`pmm.k`	Number of candidate non-missing values to sample from in the predictive mean matching steps. 0 to avoid this step.
`num.trees`	Number of trees passed to `ranger::ranger()`.
`mtry`	Number of covariates considered per split. The default `NULL` equals the rounded down root of the number of features. Can be a function, e.g., `function(p) trunc(p/3)`. Passed to `ranger::ranger()`. Note that during the first iteration, the number of features is growing. Thus, a fixed value can lead to an error. Using a function like `function(p) min(p, 2)` will fix such problem.
`min.node.size`	Minimal node size passed to `ranger::ranger()`. By default 1 for classification and 5 for regression.
`min.bucket`	Minimal terminal node size passed to `ranger::ranger()`. The default `NULL` means 1.

max.depth	Maximal tree depth passed to ranger::ranger(). NULL means unlimited depth. 1 means single split trees.
replace	Sample with replacement passed to ranger::ranger().
sample.fraction	Fraction of rows per tree passed to ranger::ranger(). The default: use all rows when replace = TRUE and 0.632 otherwise.
case.weights	Optional case weights passed to ranger::ranger().
num.threads	Number of threads passed to ranger::ranger(). The default NULL uses all threads.
save.memory	Slow but memory saving mode of ranger::ranger().
maxiter	Maximum number of iterations.
seed	Integer seed.
verbose	A value in 0, 1, 2 controlling the verbosity.
returnOOB	Should the final average OOB prediction errors be added as data attribute "oob"? Only relevant when data_only = TRUE.
data_only	If TRUE (default), only the imputed data is returned. Otherwise, a "missRanger" object with additional information is returned.
keep_forests	Should the random forests of the last relevant iteration be returned? The default is FALSE. Setting this option will use a lot of memory. Only relevant when data_only = TRUE.
...	Additional arguments passed to ranger::ranger(). Not all make sense.

Details

The iterative chaining stops as soon as maxiter is reached or if the average out-of-bag (OOB) prediction errors stop reducing. In the latter case, except for the first iteration, the second last (= best) imputed data is returned.

OOB prediction errors are quantified as 1 - R^2 for numeric variables, and as classification error otherwise. If a variable has been imputed only univariately, the value is 1.

Value

If data_only = TRUE an imputed data.frame. Otherwise, a "missRanger" object with the following elements:

- data: The imputed data.
- data_raw: The original data provided.
- forests: When keep_forests = TRUE, a list of "ranger" models used to generate the imputed data. NULL otherwise.
- to_impute: Variables to be imputed (in this order).
- impute_by: Variables used for imputation.
- best_iter: Best iteration.
- pred_errors: Per-iteration OOB prediction errors (1 - R^2 for regression, classification error otherwise).
- mean_pred_errors: Per-iteration averages of OOB prediction errors.
- pmm.k: Same as input pmm.k.

References

1. Wright, M. N. & Ziegler, A. (2016). ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R. Journal of Statistical Software, in press. <arxiv.org/abs/1508.04409>.
2. Stekhoven, D.J. and Buehlmann, P. (2012). 'MissForest - nonparametric missing value imputation for mixed-type data', Bioinformatics, 28(1) 2012, 112-118. https://doi.org/10.1093/bioinformatics/btr597.
3. Van Buuren, S., Groothuis-Oudshoorn, K. (2011). mice: Multivariate Imputation by Chained Equations in R. Journal of Statistical Software, 45(3), 1-67. http://www.jstatsoft.org/v45/i03/

Examples

```
iris2 <- generateNA(iris, seed = 1)

imp1 <- missRanger(iris2, pmm.k = 5, num.trees = 50, seed = 1)
head(imp1)

# Extended output
imp2 <- missRanger(iris2, pmm.k = 5, num.trees = 50, data_only = FALSE, seed = 1)
summary(imp2)

all.equal(imp1, imp2$data)

# Formula interface: Univariate imputation of Species and Sepal.Width
imp3 <- missRanger(iris2, Species + Sepal.Width ~ 1)
```

---

pmm *Predictive Mean Matching*

Description

For each value in the prediction vector `xtest`, one of the closest `k` values in the prediction vector `xtrain` is randomly chosen and its observed value in `ytrain` is returned. Note that `xtrain` and `xtest` must be both either numeric, logical, or factor-valued. `ytest` can be of any type.

Usage

```
pmm(xtrain, xtest, ytrain, k = 1L, seed = NULL)
```

Arguments

`xtrain`	Vector with predicted values in the training data. Must be numeric, logical, or factor-valued.
`xtest`	Vector as `xtrain` with predicted values in the test data. Missing values are not allowed.
`ytrain`	Vector of the observed values in the training data. Must be of same length as `xtrain`.
`k`	Number of nearest neighbours (donors) to sample from.
`seed`	Integer random seed.

Value

Vector of the same length as `xtest` with values from `xtrain`.

Examples

```
pmm(xtrain = c(0.2, 0.3, 0.8), xtest = c(0.7, 0.2), ytrain = 1:3, k = 1)  # c(3, 1)
```

---

predict.missRanger *Predict Method*

Description

Impute missing values on `newdata` based on an object of class "missRanger".

For multivariate imputation, use `missRanger(..., keep_forests = TRUE)`. For univariate imputation, no forests are required. This can be enforced by `predict(..., iter = 0)` or via `missRanger(. ~ 1, ...)`.

Note that out-of-sample imputation works best for rows in `newdata` with only one missing value (counting only missings in variables used as covariates in random forests). We call this the "easy case". In the "hard case", even multiple iterations (set by `iter`) can lead to unsatisfactory results.

Usage

```
## S3 method for class 'missRanger'
predict(
  object,
  newdata,
  pmm.k = object$pmm.k,
  iter = 4L,
  num.threads = NULL,
  seed = NULL,
  verbose = 1L,
  ...
)
```

Arguments

`object`	'missRanger' object.
`newdata`	A `data.frame` with missing values to impute.
`pmm.k`	Number of candidate predictions of the original dataset for predictive mean matching (PMM). By default the same value as during fitting.
`iter`	Number of iterations for "hard case" rows. 0 for univariate imputation.
`num.threads`	Number of threads used by ranger's predict function. The default `NULL` uses all threads.
`seed`	Integer seed used for initial univariate imputation and PMM.
`verbose`	Should info be printed? (1 = yes/default, 0 for no).
`...`	Passed to the predict function of ranger.

Details

The out-of-sample algorithm works as follows:

1. Impute univariately all relevant columns by randomly drawing values from the original unimputed data. This step will only impact "hard case" rows.
2. Replace univariate imputations by predictions of random forests. This is done sequentially over variables, where the variables are sorted to minimize the impact of univariate imputations. Optionally, this is followed by predictive mean matching (PMM).
3. Repeat Step 2 for "hard case" rows multiple times.

Examples

```
iris2 <- generateNA(iris, seed = 20, p = c(Sepal.Length = 0.2, Species = 0.1))
imp <- missRanger(iris2, pmm.k = 5, num.trees = 100, keep_forests = TRUE, seed = 2)
predict(imp, head(iris2), seed = 3)
```

---

print.missRanger *Print Method*

Description

Print method for an object of class "missRanger".

Usage

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

Arguments

x	An object of class "missRanger".
...	Further arguments passed from other methods.

Value

Invisibly, the input is returned.

Examples

```
CO2_ <- generateNA(CO2, seed = 1)
imp <- missRanger(CO2_, pmm.k = 5, data_only = FALSE, num.threads = 1)
imp
```

summary.missRanger *Summary Method*

Description

Summary method for an object of class "missRanger".

Usage

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

Arguments

object	An object of class "missRanger".
...	Further arguments passed from other methods.

Value

Invisibly, the input is returned.

Examples

```
CO2_ <- generateNA(CO2, seed = 1)
imp <- missRanger(CO2_, pmm.k = 5, data_only = FALSE, num.threads = 1)
summary(imp)
```

Index