Package ‘missRanger’

November 19, 2023

Title Fast Imputation of Missing Values

Version 2.4.0

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 jungle 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(). A formula interface allows to control which variables should be imputed by which.

License GPL (>= 2)

Depends R (>= 3.5.0)

Encoding UTF-8

RoxygenNote 7.2.3

Imports ranger, FNN, stats, utils

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

URL `https://github.com/mayer79/missRanger`

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

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Michael Mayer [aut, cre, cph]

Maintainer Michael Mayer <`mayermichael79@gmail.com`>

Repository CRAN

Date/Publication 2023-11-19 09:30:02 UTC

R topics documented:

`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 (see examples).
`seed`	An integer seed.

Value

`x` with missing values.

Examples

```
generateNA(1:10, p = 0.5, seed = 3345)
generateNA(rep(Sys.Date(), 10))
generateNA(cbind(1:10, 10:1), p = 0.2)
head(generateNA(iris))
head(generateNA(iris, p = 0.2))
head(generateNA(iris, p = c(0, 1, 0.5, 0.5, 0.5)))
head(generateNA(iris, p = c(Sepal.Length = 1)))
head(generateNA(iris, p = c(Species = 0.2, Sepal.Length = 0.5)))
```

`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))
imputeUnivariate(c("A", "A", NA))
imputeUnivariate(as.factor(c("A", "A", NA)))
head(imputeUnivariate(generateNA(iris)))
head(imputeUnivariate(generateNA(iris), v = "Species"))
head(imputeUnivariate(generateNA(iris), v = c("Species", "Petal.Length")))
```

`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,
  maxiter = 10L,
  seed = NULL,
  verbose = 1,
  returnOOB = FALSE,
  case.weights = NULL,
  data_only = TRUE,
  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.
`maxiter`	Maximum number of chaining iterations.
`seed`	Integer seed to initialize the random generator.
`verbose`	Controls how much info is printed to screen. 0 to print nothing. 1 (default) to print a progress bar per iteration, 2 to print the OOB prediction error per iteration and variable (1 minus R-squared for regression). Furthermore, if `verbose` is positive, the variables used for imputation are listed as well as the variables to be imputed (in the imputation order). This will be useful to detect if some variables are unexpectedly skipped.
`returnOOB`	Logical flag. If TRUE, the final average out-of-bag prediction errors per variable is added to the resulting data as attribute "oob". Only relevant when `data_only = TRUE` (and when forests are grown).
`case.weights`	Vector with non-negative case weights.
`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 final imputations be returned? The default is `FALSE`. Setting this option will use a lot of memory. Only relevant when `data_only = TRUE` (and when forests are grown).

`...`	Arguments passed to `ranger::ranger()`. If the data set is large, better use less trees (e.g. `num.trees = 20`) and/or a low value of `sample.fraction`. The following arguments are incompatible, amongst others: `write.forest`, `probability`, `split.select.weights`, `dependent.variable.name`, and `classification`.

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.

A note on `mtry`: Be careful when passing a non-default `mtry` to `ranger::ranger()` because the number of available covariates might be growing during the first iteration, depending on the missing pattern. Values `NULL` (default) and 1 are safe choices. Additionally, recent versions of `ranger::ranger()` allow `mtry` to be a single-argument function of the number of available covariables, e.g., `mtry = function(m) max(1, m %/% 3)`.

Value

If `data_only` an imputed `data.frame`. Otherwise, a "missRanger" object with the following elements that can be extracted via `$`:

- `data`: The imputed data.
- `forests`: When `keep_forests = TRUE`, a list of "ranger" models used to generate the imputed data. `NULL` otherwise.
- `visit_seq`: 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.

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

```
irisWithNA <- generateNA(iris, seed = 34)
irisImputed <- missRanger(irisWithNA, pmm.k = 3, num.trees = 100)
head(irisImputed)
head(irisWithNA)

## Not run:
# Extended output
imp <- missRanger(irisWithNA, pmm.k = 3, num.trees = 100, data_only = FALSE)
head(imp$data)
imp$pred_errors

# If you even want to keep the random forests of the best iteration
imp <- missRanger(
  irisWithNA, pmm.k = 3, num.trees = 100, data_only = FALSE, keep_forests = TRUE
)
imp$forests$Sepal.Width
imp$pred_errors[imp$best_iter, "Sepal.Width"]  # 1 - R-squared

## End(Not run)
```

---

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.

Usage

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

Arguments

`xtrain`	Vector with predicted values in the training data. Can be of type logical, numeric, character, or factor.
`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`. Missing values in either of `xtrain` or `ytrain` will be dropped in a pairwise manner.
`k`	Number of nearest neighbours 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, 0.2, 0.8), xtest = 0.3, ytrain = c(0, 0, 1)) # 0
pmm(xtrain = c(TRUE, FALSE, TRUE), xtest = FALSE, ytrain = c(2, 0, 1)) # 0
pmm(xtrain = c(0.2, 0.8), xtest = 0.3, ytrain = c("A", "B"), k = 2) # "A" or "B"
pmm(xtrain = c("A", "A", "B"), xtest = "A", ytrain = c(2, 2, 4), k = 2) # 2
pmm(xtrain = factor(c("A", "B")), xtest = factor("C"), ytrain = 1:2) # 2
```

`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