Package ‘missRanger’

June 30, 2019

Title Fast Imputation of Missing Values

Version 2.1.0

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.

Depends R (>= 3.5.0)

License GPL(>= 2)

Encoding UTF-8

LazyData true

Type Package

Date 2019-06-30

Imports stats, FNN (>= 1.1), ranger (>= 0.10)

Suggests mice, dplyr, survival, ggplot2, knitr, rmarkdown

VignetteBuilder knitr

Author Michael Mayer [aut, cre, cph]

Maintainer Michael Mayer <mayermichael79@gmail.com>

RoxygenNote 6.1.1

NeedsCompilation no

R topics documented:

`convert` *Conversion of non-factor/non-numeric variables.*

Description

Converts non-factor/non-numeric variables in a data frame to factor/numeric. Stores information to revert back.

Usage

```
convert(X, check = FALSE)
```

Arguments

`X` A data frame.

`check` If `TRUE`, the function checks if the converted columns can be reverted without changes.

Value

A list with the following elements: `X` is the converted data frame, `vars`, `types`, `classes` are the names, types and classes of the converted variables. Finally, `bad` names variables in `X` that should have been converted but could not.

Author(s)

Michael Mayer

`generateNA` *Adds Missing Values to a Vector, Matrix or Data Frame*

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.

Author(s)

Michael Mayer

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.

Author(s)

Michael Mayer

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` *missRanger*

Description

Uses the "ranger" package [1] to do fast missing value imputation by chained random forests, see [2] and [3]. 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 and, as such, allows to do multiple imputation when repeating the call to missRanger(). The iterative chaining stops as soon as `maxiter` is reached or if the average out-of-bag estimate of performance stops improving. In the latter case, except for the first iteration, the second last (i.e. best) imputed data is returned.

Usage

```
missRanger(data, formula = . ~ ., pmm.k = 0L, maxiter = 10L,
  seed = NULL, verbose = 1, returnOOB = FALSE, case.weights = NULL,
  ...)
```

Arguments

`data` A `data.frame` or `tibble` 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. If e.g. 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 step. 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 "." per iteration and variable, 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 usedful to detect if some variables are unexpectedly skipped.

`returnOOB` Logical flag. If TRUE, the final average out-of-bag prediction error is added to the output as attribute "oob". This does not work in the special case when the variables are imputed univariately.

`case.weights` Vector with non-negative case weights.

`...` Arguments passed to `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 e.g. incompatible with `ranger`: `mtry`, `write.forest`, `probability`, `split.select.weights`, `dependent.variable.name`, and `classification`.

Details

Fast Imputation of Missing Values by Chained Random Forests

Value

An imputed `data.frame`.

Author(s)

Michael Mayer

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. http://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:
# With extra trees algorithm
irisImputed_et <- missRanger(irisWithNA, pmm.k = 3, num.trees = 100, splitrule = "extratrees")
head(irisImputed_et)

# Do not impute Species. Note: Since this variable contains missings, it won't be used
# for imputing other variables.
head(irisImputed <- missRanger(irisWithNA, . - Species ~ ., pmm.k = 3, num.trees = 100))

# Impute univariately only.
head(irisImputed <- missRanger(irisWithNA, . ~ 1))

# Use Species and Petal.Length to impute Species and Petal.Length.
head(irisImputed <- missRanger(irisWithNA, Species + Petal.Length ~ Species + Petal.Length,
                               pmm.k = 3, num.trees = 100))

# Multiple imputation: Fill data 20 times, run 20 analyses and pool their results.
require(mice)
filled <- replicate(20, missRanger(irisWithNA, verbose = 0, num.trees = 100, pmm.k = 5),
                    simplify = FALSE)
models <- lapply(filled, function(x) lm(Sepal.Length ~ ., x))
summary(pooled_fit <- pool(models)) # Realistically inflated standard errors and p values

# A data set with logicals, numerics, characters and factors.
n <- 100
X <- data.frame(x1 = seq_len(n),
                x2 = log(seq_len(n)),
                x3 = sample(LETTERS[1:3], n, replace = TRUE),
                x4 = factor(sample(LETTERS[1:3], n, replace = TRUE)),
                x5 = seq_len(n) > 50)
head(X)
X_NA <- generateNA(X, p = seq(0, 0.8, by = .2))
head(X_NA)

head(X_imp <- missRanger(X_NA))
head(X_imp <- missRanger(X_NA, pmm = 3))
head(X_imp <- missRanger(X_NA, pmm = 3, verbose = 0))
head(X_imp <- missRanger(X_NA, pmm = 3, verbose = 2, returnOOB = TRUE))
attr(X_imp, "oob") # OOB prediction errors per column.

# The formula interface
head(X_imp <- missRanger(X_NA, x2 ~ x2 + x3, pmm = 3)) # Does not use x3 because of NAs
head(X_imp <- missRanger(X_NA, x2 + x3 ~ x2 + x3, pmm = 3))
head(X_imp <- missRanger(X_NA, x2 + x3 ~ 1, pmm = 3)) # Univariate imputation

## End(Not run)
```

pmm *missRanger pmm*

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.

Details

Predictive Mean Matching

Value

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

Author(s)

Michael Mayer

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
```

`revert` *Revert conversion.*

Description

Reverts conversions done by `convert`.

Usage

```
revert(con, X = con$X)
```

Arguments

`con`	A list returned by `convert`.
`X`	A data frame with some columns to be converted back according to the information stored in `converted`.

Value

A data frame.

Author(s)

Michael Mayer

`typeof2` *A version of* `typeof` *internally used by* `missRanger`.

Description

Returns either "numeric" (double or integer), "factor", "character", "logical", "special" (mode numeric, but neither double nor integer) or "" (otherwise). `missRanger` requires this information to deal with response types not natively supported by `ranger`.

Usage

```
typeof2(object)
```

Arguments

`object`	Any object.

Value

A string.

Author(s)

Michael Mayer

Index