Package 'missRanger'

August 24, 2017

Title missRanger - Imputation by Chained Random Forests

Version 0.1.2

Description Fast version of the \{}code{missForest} package based on \{}code{ranger} (a fast implementation of random forests called 'random jungle'). We offer the option of predictive mean matching to add extra variability regarding multiple imputations and to avoid imputation by values not already being present in the data.

Depends R (>= 3.3.2)

License GPL-3

Encoding UTF-8

LazyData true

Imports stats, FNN, ranger (>= 0.6.0)

RoxygenNote 6.0.1

NeedsCompilation no

Author Michael Mayer [aut, cre]

Maintainer Michael Mayer <mayermichael79@gmail.com>

R topics documented:

`generateNA` *Adds Missing Values to a Data Set*

Description

Takes a data frame and replaces randomly part of the values by missing values.

Usage

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

Arguments

data	A data.frame.
p	Proportion of missing values to approximatly add to each column of data.
seed	An integer seed.

Value

data with missing values.

Examples

```
head(generateNA(iris))
```

---

imputeUnivariate *Univariate Imputation*

Description

Fills missing values of a vector of any type by sampling with replacement from the non-missing values. Requires at least one non-missing value to run.

Usage

```
imputeUnivariate(x)
```

Arguments

x	A vector of any type possibly containing missing values.

Value

A vector of the same length and type as x but without missing values.

Examples

```
imputeUnivariate(c(NA, 0, 1, 0, 1))
imputeUnivariate(c("A", "A", NA))
```

`missRanger` *Missing Values Imputation by Chained Random Forests*

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, we offer the option of using predictive mean matching. This firstly avoids the imputation with values not 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'.

Usage

```
missRanger(data, maxiter = 10L, pmm.k = 0L, seed = NULL, ...)
```

Arguments

`data`	A `data.frame` with missing values to impute.
`maxiter`	Maximum number of chaining iterations.
`pmm.k`	Number of candidate non-missing values to sample from in the predictive mean matching step. Use `pmm.k = 0` to avoid this step.
`seed`	Integer seed to initialize the random generator.
`...`	Arguments passed to `ranger`. Don't use `formula`, `data` or `seed`. They are already handled by the algorithm. Not all `ranger` options do make sense (e.g. `write.forest = FALSE` will cause the algorithm to crash. If the data set is large, better use less trees `num.trees = 100` and/or a low value of `sample.fraction`.

Value

A `data.frame` as `data` but with imputed missing values.

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, doi: 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)
irisImputed <- missRanger(irisWithNA, pmm.k = 3)
head(irisImputed)
head(irisWithNA)
head(iris)
```

pmm *Predictive Mean Matching*

Description

This function is used internally only but might help others to implement an efficient way of doing predictive mean matching on top of any prediction based missing value imputation. It works as follows: For each predicted value of a vector `xtest`, the closest `k` predicted values of another vector `xtrain` are identified by k-nearest neighbour. Then, one of those neighbours are randomly picked and its corresponding 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 set.
`xtest`	Vector with predicted values in the test data set.
`ytrain`	Vector with observed response in the training data set.
`k`	Number of nearest neighbours to choose from. Set `k = 0` if no predictive mean matching is to be done.
`seed`	Integer random seed.

Value

Vector with predicted values in the test data set based on preditive mean matching.

Examples

```
pmm(xtrain = c(0.2, 0.2, 0.8), xtest = 0.3, ytrain = c(0, 0, 1), k = 1) # 0
pmm(xtrain = c(0.2, 0.2, 0.8), xtest = 0.3, ytrain = c(0, 0, 1), k = 3) # 0 or 1
pmm(xtrain = c("A", "A", "B"), xtest = "B", ytrain = c("B", "A", "B"), k = 1) # B
pmm(xtrain = c("A", "A", "B"), xtest = "B", ytrain = c("B", "A", "B"), k = 2) # A or B
```

Index