Package 'CALIBERrfimpute'

May 7, 2026

Type Package

Title Multiple Imputation Using MICE and Random Forest

Version 1.0-8

Date 2026-02-17

Description Functions to impute using random forest under full conditional specifications (multivariate imputation by chained equations). The methods are described in Shah and others (2014) <doi:10.1093/aje/kwt312>.

License GPL-3

Depends mice (>= 2.20)

Imports mvtnorm, randomForest

Suggests missForest, rpart, survival, xtable, ranger

Author Anoop Shah [aut, cre],
Jonathan Bartlett [ctb],
Harry Hemingway [ths],
Owen Nicholas [ths],
Aroon Hingorani [ths]

Maintainer Anoop Shah <anoop@doctors.org.uk>

BuildVignettes TRUE

NeedsCompilation no

Repository CRAN

Date/Publication 2026-02-23 13:10:10 UTC

Contents

CALIBERrfimpute-package

Imputation in MICE using Random Forest

Description

Multivariate Imputation by Chained Equations (MICE) is commonly used to impute missing values in analysis datasets using full conditional specifications. However, it requires that the predictor models are specified correctly, including interactions and nonlinearities. Random Forest is a regression and classification method which can accommodate interactions and non-linearities without requiring a particular statistical model to be specified.

Details

CALIBERrfimpute provides functions that can be used as imputation functions with `mice`. Note that the mice package itself provides the `mice.impute.rf` function for imputation using Random Forest, as of version 2.20. The CALIBERrfimpute package provides different, independently developed imputation functions using Random Forest in MICE.

This package contains reports of two simulation studies:

Simulation study (linear regression): A comparison of Random Forest and parametric MICE in a linear regression example. To open this report, run: `utils::browseURL(system.file("doc", "simstudy.pdf", package = "CALIBERrfimpute"))`

Survival analysis with interactions (large sample): Compares the Random Forest MICE algorithm (`mice.impute.rfcont`) with parametric MICE and the algorithms of Doove et al. (`mice.impute.cart` and `mice.impute.rf`). To open this report, run: `utils::browseURL(system.file("doc", "simstudy_survival_largesample.pdf", package = "CALIBERrfimpute"))`

Survival analysis with interactions (small sample): A small sample version of the above simulation as a vignette. See `vignette("simstudy_survival", package = "CALIBERrfimpute")`

Package:	CALIBERrfimpute
Type:	Package
Version:	1.0-8
Date:	2026-02-17
License:	GPL-3

Author(s)

Anoop Shah

Maintainer: anoop@doctors.org.uk

References

Shah AD, Bartlett JW, Carpenter J, Nicholas O, Hemingway H. Comparison of Random Forest and parametric imputation models for imputing missing data using MICE: a CALIBER study. American Journal of Epidemiology 2014; 179(6): 764–774. doi: 10.1093/aje/kwt312

Doove LL, van Buuren S, Dusseldorp E. Recursive partitioning for missing data imputation in the presence of interaction effects. Computational Statistics and Data Analysis 2014; 72: 92–104. doi: 10.1016/j.csda.2013.10.025

See Also

mice, randomForest, mice.impute.rfcont, mice.impute.rfcat, mice.impute.rf

---

makemar *Creates artificial missing at random missingness*

Description

Introduces missingness into x1 and x2 into a data.frame of the format produced by simdata, for use in the simulation study. The probability of missingness depends on the logistic of the fully observed variables y and x3; hence it is missing at random but not missing completely at random.

Usage

```
makemar(simdata, prop = 0.2)
```

Arguments

simdata	simulated dataset created by simdata.
prop	proportion of missing values to be introduced in x1 and x2.

Details

This function is used for simulation and testing.

Value

A data.frame with columns:

y	dependent variable, based on the model y = x1 + x2 + x3 + normal error
x1	partially observed continuous variable
x2	partially observed continuous or binary (factor) variable
x3	fully observed continuous variable
x4	variable not in the model to predict y, but associated with x1, x2 and x3; used as an auxiliary variable in imputation

See Also

simdata

Examples

```
set.seed(1)
mydata <- simdata(n=100)
mymardata <- makemar(mydata, prop=0.1)
# Count the number of missing values
sapply(mymardata, function(x){sum(is.na(x))})
# y x1 x2 x3 x4
# 0 11 10  0  0
```

---

mice.impute.rfcat *Impute categorical variables using Random Forest within MICE*

Description

This method can be used to impute logical or factor variables (binary or >2 levels) in MICE by specifying method = 'rfcat'. It was developed independently from the mice.impute.rf algorithm of Doove et al., and differs from it in some respects.

Usage

```
mice.impute.rfcat(y, ry, x, ntree_cat = NULL,
    nodesize_cat = NULL, maxnodes_cat = NULL, ntree = NULL, ...)
```

Arguments

y	a logical or factor vector of observed values and missing values of the variable to be imputed.
ry	a logical vector stating whether y is observed or not.
x	a matrix of predictors to impute y.
ntree_cat	number of trees, default = 10. A global option can be set thus: setRFoptions(ntree_cat=10).
nodesize_cat	minimum size of nodes, default = 1. A global option can be set thus: setRFoptions(nodesize_cat=1). Smaller values of nodesize create finer, more precise trees but increase the computation time.
maxnodes_cat	maximum number of nodes, default NULL. If NULL the number of nodes is determined by number of observations and nodesize_cat.
ntree	an alternative argument for specifying the number of trees, over-ridden by ntree_cat. This is for consistency with the mice.impute.rf function.
...	other arguments to pass to randomForest.

Details

This Random Forest imputation algorithm has been developed as an alternative to logistic or polytomous regression, and can accommodate non-linear relations and interactions among the predictor variables without requiring them to be specified in the model. The algorithm takes a bootstrap sample of the data to simulate sampling variability, fits a set of classification trees, and chooses each imputed value as the prediction of a randomly chosen tree.

Value

A vector of imputed values of y.

Note

This algorithm has been tested on simulated data and in survival analysis of real data with artificially introduced missingness completely at random. There was slight bias in hazard ratios compared to polytomous regression, but coverage of confidence intervals was correct.

Author(s)

Anoop Shah

References

Shah AD, Bartlett JW, Carpenter J, Nicholas O, Hemingway H. Comparison of Random Forest and parametric imputation models for imputing missing data using MICE: a CALIBER study. American Journal of Epidemiology 2014; 179(6): 764–774. doi: 10.1093/aje/kwt312

See Also

setRFoptions, mice.impute.rfcont, mice, mice.impute.rf, mice.impute.cart, randomForest

Examples

```
set.seed(1)

# A small sample dataset
mydata <- data.frame(
    x1 = as.factor(c('this', 'this', NA, 'that', 'this')),
    x2 = 1:5,
    x3 = c(TRUE, FALSE, TRUE, NA, FALSE))
mice(mydata, method = c('logreg', 'norm', 'logreg'), m = 2, maxit = 2)
mice(mydata[, 1:2], method = c('rfcat', 'rfcont'), m = 2, maxit = 2)
mice(mydata, method = c('rfcat', 'rfcont', 'rfcat'), m = 2, maxit = 2)

# A larger simulated dataset
mydata <- simdata(100, x2binary = TRUE)
mymardata <- makemar(mydata)

cat('\nNumber of missing values:\n')
print(sapply(mymardata, function(x){sum(is.na(x))}))
```

```
# Test imputation of a single column in a two-column dataset
cat('\nTest imputation of a simple dataset')
print(mice(mymardata[, c('y', 'x2')], method = 'rfcat', m = 2, maxit = 2))

# Analyse data
cat('\nFull data analysis:\n')
print(summary(lm(y ~ x1 + x2 + x3, data = mydata)))

cat('\nMICE normal and logistic:\n')
print(summary(pool(with(mice(mymardata,
    method = c('', 'norm', 'logreg', '', ''), m = 2, maxit = 2),
    lm(y ~ x1 + x2 + x3)))))

# Set options for Random Forest
setRFoptions(ntree_cat = 10)

cat('\nMICE using Random Forest:\n')
print(summary(pool(with(mice(mymardata,
    method = c('', 'rfcont', 'rfcat', '', ''), m = 2, maxit = 2),
    lm(y ~ x1 + x2 + x3)))))

cat('\nDataset with unobserved levels of a factor\n')
data3 <- data.frame(x1 = 1:100, x2 = factor(c(rep('A', 25),
    rep('B', 25), rep('C', 25), rep('D', 25))))
data3$x2[data3$x2 == 'D'] <- NA
mice(data3, method = c('', 'rfcat'), m = 2, maxit = 2)
```

`mice.impute.rfcont` *Impute continuous variables using Random Forest within MICE*

Description

This method can be used to impute continuous variables in MICE by specifying method = 'rfcont'. It was developed independently from the `mice.impute.rf` algorithm of Doove et al., and differs from it in drawing imputed values from a normal distribution.

Usage

```
mice.impute.rfcont(y, ry, x, ntree_cont = NULL,
    nodesize_cont = NULL, maxnodes_cont = NULL, ntree = NULL, ...)
```

Arguments

`y`	a vector of observed values and missing values of the variable to be imputed.
`ry`	a logical vector stating whether y is observed or not.
`x`	a matrix of predictors to impute y.
`ntree_cont`	number of trees, default = 10. A global option can be set thus: `setRFoptions(ntree_cont=10)`.

nodesize_cont	minimum size of nodes, default = 5. A global option can be set thus: `setRFoptions(nodesize_cont=5)`. Smaller values of nodesize create finer, more precise trees but increase the computation time.
maxnodes_cont	maximum number of nodes, default NULL. If NULL the number of nodes is determined by number of observations and nodesize_cont.
ntree	an alternative argument for specifying the number of trees, over-ridden by `ntree_cont`. This is for consistency with the `mice.impute.rf` function.
...	other arguments to pass to randomForest.

Details

This Random Forest imputation algorithm has been developed as an alternative to normal-based linear regression, and can accommodate non-linear relations and interactions among the predictor variables without requiring them to be specified in the model. The algorithm takes a bootstrap sample of the data to simulate sampling variability, fits a regression forest trees and calculates the out-of-bag mean squared error. Each value is imputed as a random draw from a normal distribution with mean defined by the Random Forest prediction and variance equal to the out-of-bag mean squared error.

If only one tree is used (not recommended), a bootstrap sample is not taken in the first stage because the Random Forest algorithm performs an internal bootstrap sample before fitting the tree.

Value

A vector of imputed values of y.

Note

This algorithm has been tested on simulated data with linear regression, and in survival analysis of real data with artificially introduced missingness at random. On the simulated data there was slight bias if the distribution of missing values was very different from observed values, because imputed values were closer to the centre of the data than the missing values. However in the survival analysis the hazard ratios were unbiased and coverage of confidence intervals more conservative than normal-based MICE, but the mean length of confidence intervals was shorter with mice.impute.rfcont.

Author(s)

Anoop Shah

References

Shah AD, Bartlett JW, Carpenter J, Nicholas O, Hemingway H. Comparison of Random Forest and parametric imputation models for imputing missing data using MICE: a CALIBER study. American Journal of Epidemiology 2014; 179(6): 764–774. doi: 10.1093/aje/kwt312

See Also

`setRFoptions`, `mice.impute.rfcat`, `mice`, `mice.impute.rf`, `mice.impute.cart`, `randomForest`

Examples

```
set.seed(1)

# A small dataset with a single row to be imputed
mydata <- data.frame(x1 = c(2, 3, NA, 4, 5, 1, 6, 8, 7, 9), x2 = 1:10,
    x3 = c(1, 3, NA, 4, 2, 8, 7, 9, 6, 5))
mice(mydata, method = c('norm', 'norm', 'norm'), m = 2, maxit = 2)
mice(mydata[, 1:2], method = c('rfcont', 'rfcont'), m = 2, maxit = 2)
mice(mydata, method = c('rfcont', 'rfcont', 'rfcont'), m = 2, maxit = 2)

# A larger simulated dataset
mydata <- simdata(100)
cat('\nSimulated multivariate normal data:\n')
print(data.frame(mean = colMeans(mydata), sd = sapply(mydata, sd)))

# Apply missingness pattern
mymardata <- makemar(mydata)
cat('\nNumber of missing values:\n')
print(sapply(mymardata, function(x){sum(is.na(x))}))

# Test imputation of a single column in a two-column dataset
cat('\nTest imputation of a simple dataset')
print(mice(mymardata[, c('y', 'x1')], method = 'rfcont'))

# Analyse data
cat('\nFull data analysis:\n')
print(summary(lm(y ~ x1 + x2 + x3, data=mydata)))

cat('\nMICE using normal-based linear regression:\n')
print(summary(pool(with(mice(mymardata,
    method = 'norm'), lm(y ~ x1 + x2 + x3)))))

# Set options for Random Forest
setRFoptions(ntree_cont = 10)

cat('\nMICE using Random Forest:\n')
print(summary(pool(with(mice(mymardata,
    method = 'rfcont'), lm(y ~ x1 + x2 + x3)))))
```

`setRFoptions` *Set Random Forest options for imputation using MICE*

Description

A convenience function to set global options for number of trees or number of nodes.

Usage

```
setRFoptions(ntree_cat = NULL, ntree_cont = NULL,
    nodesize_cat = NULL, nodesize_cont = NULL,
    maxnodes_cat = NULL, maxnodes_cont = NULL)
```

Arguments

ntree_cat	number of trees to be used for imputing categorical variables (each imputed value is the prediction of a randomly chosen tree), default = 10.
ntree_cont	number of trees in the forest for imputing continuous variables, default = 10.
nodesize_cat	minimum node size for trees for imputing categorical variables, default = 1. A higher value can be used on larger datasets in order to save time.
nodesize_cont	minimum node size for trees for imputing continuous variables, default = 5. A higher value can be used on larger datasets in order to save time.
maxnodes_cat	maximum number of nodes in trees for imputing categorical variables. By default the size limit is set by the number of observations and nodesize_cat.
maxnodes_cont	maximum number of nodes in trees for imputing continuous variables. By default the size limit is set by the number of observations and nodesize_cont.

Details

This function sets the global options which have the prefix 'CALIBERrfimpute_'.

Value

No return value. The function prints a message stating the new option setting.

See Also

mice.impute.rfcat, mice.impute.rfcont

Examples

```
# Set option using setRFoptions
setRFoptions(ntree_cat=15)
options()$CALIBERrfimpute_ntree_cat

# Set option directly
options(CALIBERrfimpute_ntree_cat=20)
options()$CALIBERrfimpute_ntree_cat
```

---

simdata *Simulate multivariate data for testing*

Description

Creates multivariate normal or normal and binary data, as used in the simulation study.

Usage

```
simdata(n = 2000, mymean = rep(0, 4), mysigma = matrix(
c( 1,   0.2, 0.1, -0.7,
   0.2, 1,   0.3,  0.1,
   0.1, 0.3, 1,    0.2,
  -0.7, 0.1, 0.2,  1), byrow = TRUE, nrow = 4, ncol = 4),
residsd = 1, x2binary = FALSE)
```

Arguments

n	number of observations to create.
mymean	vector of length 4, giving the mean of each variable.
mysigma	variance-covariance matrix of multivariate normal distribution from which x1-x4 are to be drawn.
residsd	residual standard deviation.
x2binary	if TRUE, x2 is converted to a binary factor variable (1, 2) with probability equal to the logistic of the underlying normally distributed variable.

Value

Data frame with 5 columns:

y	continuous, generated by y = x1 + x2 + x3 + normal error if x2 is continuous, or y = x1 + x2 + x3 - 1 + normal error if x2 is a factor with values 1 or 2
x1	continuous
x2	continuous or binary (factor) with value 1 or 2
x3	continuous
x4	continuous

See Also

makemar

Examples

```
set.seed(1)
simdata(n=4, x2binary=TRUE)
#             y          x1 x2         x3        x4
# 1 -0.06399616 -1.23307320  2 -0.6521442 1.6141842
# 2  1.00822173 -0.05167026  1  0.4659907 0.5421826
# 3  2.87886825  0.43816687  1  1.5217240 0.2808691
# 4  0.79129101 -0.72510640  1  0.7342611 0.1820001
```

Index