Package ‘miceRanger’

October 13, 2022

Title Multiple Imputation by Chained Equations with Random Forests

Version 1.5.0

Maintainer Sam Wilson <samwilson303@gmail.com>

Description Multiple Imputation has been shown to
be a flexible method to impute missing values by
Van Buuren (2007) <doi:10.1177/0962280206074463>.
Expanding on this, random forests have been shown
to be an accurate model by Stekhoven and Buhlmann
<arXiv:1105.0828> to impute missing values in datasets.
They have the added benefits of returning out of bag
error and variable importance estimates, as well as
being simple to run in parallel.

URL https://github.com/FarrellDay/miceRanger

BugReports https://github.com/FarrellDay/miceRanger/issues

Encoding UTF-8

LazyData true

License MIT + file LICENSE

Depends R (>= 3.5.0)

Imports ranger, data.table, stats, FNN, ggplot2, crayon, corrplot,
ggpubr, DescTools, foreach

Suggests knitr, rmarkdown, doParallel, testthat (>= 2.1.0)

RoxygenNote 7.1.1

VignetteBuilder knitr

NeedsCompilation no

Author Sam Wilson [aut, cre]

Repository CRAN

Date/Publication 2021-09-06 15:20:02 UTC

R topics documented:

`addDatasets` *addDatasets*

Description

Add datasets to a current miceDefs object. Adds the same number of iterations as other datasets.

Usage

```
addDatasets(miceObj, datasets = 3, parallel = FALSE, verbose = TRUE, ...)
```

Arguments

`miceObj`	A miceDefs object created by `miceRanger`.
`datasets`	The number of datasets to add.
`parallel`	Should the process run in parallel? This process will take advantage of any cluster set up when `miceRanger` is called.
`verbose`	should progress be printed?
`...`	other parameters passed to `ranger()` to control model building.

Value

an updated miceDefs object with additional datasets.

Examples

```
data("sampleMiceDefs")
miceObj <- addDatasets(
    sampleMiceDefs
  , datasets = 1
  , verbose = FALSE
  , num.threads = 1
  , num.trees=5
)
```

`addIterations` *addIterations*

Description

Add iterations to a current miceDefs object. Adds iterations for all datasets.

Usage

```
addIterations(miceObj, iters = 5, parallel = FALSE, verbose = TRUE, ...)
```

Arguments

`miceObj`	A miceDefs object created by `miceRanger`.
`iters`	The number of iterations to add to each dataset.
`parallel`	Should the process run in parallel? This process will take advantage of any cluster set up when `miceRanger` is called.
`verbose`	should progress be printed?
`...`	other parameters passed to `ranger()` to control model building.

Value

an updated miceDefs object with additional iterations.

Examples

```
data("sampleMiceDefs")
miceObj <- addIterations(
    sampleMiceDefs
  , iters=2
  , verbose=FALSE
  , num.threads = 1
  , num.trees=5
)
```

`amputeData` *amputeData*

Description

Randomly amputes data (MCAR).

Usage

```
amputeData(data, perc = 0.1, cols = names(data))
```

Arguments

`data`	The data to be amputed
`perc`	A scalar. The percentage (0-1) to be amputed.
`cols`	The columns to ampute.

Value

The same dataset with random values in `cols` set to NA.

Examples

```
data(iris)
head(iris,10)

ampIris <- amputeData(iris)
head(ampIris,10)
```

`completeData` *completeData*

Description

Return imputed datasets from a `miceDefs` object.

Usage

```
completeData(miceObj, datasets = 1:miceObj$callParams$m, verbose = TRUE)
```

Arguments

`miceObj`	an object of class miceDefs.
`datasets`	a vector of the datasets you want to return.
`verbose`	a warning is thrown if integers are converted to doubles. To suppress this warning, set to `FALSE`.

Value

A list of imputed datasets.

Examples

```
data("sampleMiceDefs")
imputedList <- completeData(sampleMiceDefs)
```

`getVarImps` *Get Variable Imputations*

Description

Returns imputations for the specified datasets and variable.

Usage

```
getVarImps(x, datasets, var)
```

Arguments

`x`	A `miceDefs` or `impDefs` object.
`datasets`	The datasets to return. Can be a number, of a numeric vector.
`var`	The variable to return the imputations for.

Details

These functions exist solely to get at the imputed data for a specific dataset and variable.

Value

A matrix of imputations for a single variable. Each column represents a different dataset.

Examples

```
data("sampleMiceDefs")
getVarImps(sampleMiceDefs,var="Petal.Width")
```

impute — *Impute New Data With Existing Models*

Description

Impute data using the information from an existing `miceDefs` object.

Usage

```
impute(
  data,
  miceObj,
  datasets = 1:miceObj$callParams$m,
  iterations = miceObj$callParams$maxiter,
  verbose = TRUE
)
```

Arguments

`data`	The data to be imputed. Must have all columns used in the imputation of miceDefs.
`miceObj`	A miceDefs object created by `miceRanger()`.
`datasets`	A numeric vector specifying the datasets with which to impute `data`. See details for more information.
`iterations`	The number of iterations to run. By default, the same as the number of iterations currently in `miceObj`.
`verbose`	should progress be printed?

Details

This capability is experimental, but works well in benchmarking. The original data and random forests (if returnModels = TRUE) are returned when `miceRanger` is called. These models can be recycled to impute a new dataset in the same fashion as `miceRanger`, by imputing each variable over a series of iterations. Each dataset created in `miceObj` can be thought of as a different imputation mechanism, with different initialized values and a different associated random forests. Therefore, it is necessary to choose the datasets which will be used to impute the data. When mean matching a numeric variable, the candidate values are drawn from the original data passed to `miceRanger`, not the `data` passed to this function.

Value

An object of class impDefs, which contains information about the imputation process.

`callParams`	The parameters of the object.
`data`	The original data provided by the user.
`naWhere`	Logical index of missing data, having the same dimensions as `data`.

`missingCounts`	The number of missing values for each variable.
`imputedData`	A list of imputed datasets.

Examples

```
ampDat <- amputeData(iris)
miceObj <- miceRanger(ampDat,1,1,returnModels=TRUE,verbose=FALSE)

newDat <- amputeData(iris)
newImps <- impute(newDat,miceObj)
```

`miceRanger` *miceRanger: Fast Imputation with Random Forests*

Description

Performs multiple imputation by chained random forests. Returns a miceDefs object, which contains information about the imputation process.

Usage

```
miceRanger(
  data,
  m = 5,
  maxiter = 5,
  vars,
  valueSelector = c("meanMatch", "value"),
  meanMatchCandidates = pmax(round(nrow(data) * 0.01), 5),
  returnModels = FALSE,
  parallel = FALSE,
  verbose = TRUE,
  ...
)
```

Arguments

`data`	A data.frame or data.table to be imputed.
`m`	The number of datasets to produce.
`maxiter`	The number of iterations to run for each dataset.
`vars`	Specifies which and how variables should be imputed. Can be specified in 3 different ways: • <missing> If not provided, all columns will be imputed using all columns. If a column contains no missing values, it will still be used as a feature to impute missing columns.

- <character vector> If a character vector of column names is passed, these columns will be imputed using all available columns in the dataset. The order of this vector will determine the order in which the variables are imputed.
- <named list of character vectors> Predictors can be specified for each variable with a named list. List names are the variables to impute. Elements in the vectors should be features used to impute that variable. The order of this list will determine the order in which the variables are imputed.

`valueSelector`	How to select the value to be imputed from the model predictions. Can be "meanMatching", "value", or a named vector containing a mixture of those values. If a named vector is passed, the names must equal the variables to be imputed specified in `vars`.
`meanMatchCandidates`	Specifies the number of candidate values which are selected from in the mean matching algorithm. Can be either specified as an integer or a named integer vector for different values by variable. If a named integer vector is passed, the names of the vector must contain at a minimum the names of the numeric variables imputed using `valueSelector = "meanMatch"`.
`returnModels`	Logical. Should the final model for each variable be returned? Set to `TRUE` to use the `impute` function, which allows imputing new samples without having to run `miceRanger` again. Setting to TRUE can cause the returned `miceDefs` object to take up a lot of memory. Use only if you plan on using the `impute` function.
`parallel`	Should the process run in parallel? Usually not necessary. This process will take advantage of any cluster set up when `miceRanger` is called.
`verbose`	should progress be printed?
`...`	other parameters passed to `ranger()` to control forest growth.

Value

a miceDefs object, containing the following:

`callParams`	The parameters of the object.
`data`	The original data provided by the user, cast to a data.table.
`naWhere`	Logical index of missing data, having the same dimensions as `data`.
`missingCounts`	The number of missing values for each variable
`rawClasses`	The original classes provided in `data`
`newClasses`	The new classes of the returned data.
`allImps`	The imputations of all variables at each iteration, for each dataset.
`allImport`	The variable importance metrics at each iteration, for each dataset.
`allError`	The OOB model error for all variables at each iteration, for each dataset.
`finalImps`	The final imputations for each dataset.
`finalImport`	The final variable importance metrics for each dataset.
`finalError`	The final model error for each variable in every dataset.

`finalModels`	Only returned if `returnModels = TRUE`. A list of `ranger` random forests for each dataset/variable.
`imputationTime`	The total time in seconds taken to create the imputations for the specified datasets and iterations. Does not include any setup time.

Vignettes

It is highly recommended to visit the GitHub README for a thorough walkthrough of miceRanger's capabilities, as well as performance benchmarks.

Several vignettes are also available on miceRanger's listing on the CRAN website.

1. The MICE Algorithm
2. Imputing Missing Data with miceRanger
3. Diagnostic Plotting

Examples

```
#################
## Simple Example

data(iris)
ampIris <- amputeData(iris)

miceObj <- miceRanger(
    ampIris
  , m = 1
  , maxiter = 1
  , verbose=FALSE
  , num.threads = 1
  , num.trees=5
)


##################
## Run in parallel

data(iris)
ampIris <- amputeData(iris)

library(doParallel)
cl <- makeCluster(2)
registerDoParallel(cl)

# Perform mice
miceObjPar <- miceRanger(
    ampIris
  , m = 2
  , maxiter = 2
  , parallel = TRUE
  , verbose = FALSE
)
```

```
stopCluster(cl)
registerDoSEQ()


############################
## Complex Imputation Schema

data(iris)
ampIris <- amputeData(iris)

# Define variables to impute, as well as their predictors
v <- list(
  Sepal.Width = c("Sepal.Length","Petal.Width","Species")
  , Sepal.Length = c("Sepal.Width","Petal.Width")
  , Species = c("Sepal.Width")
)

# Specify mean matching for certain variables.
vs <- c(
  Sepal.Width = "meanMatch"
  , Sepal.Length = "value"
  , Species = "meanMatch"
)

# Different mean matching candidates per variable.
mmc <- c(
  Sepal.Width = 4
  , Species = 10
)

miceObjCustom <- miceRanger(
    ampIris
  , m = 1
  , maxiter = 1
  , vars = v
  , valueSelector = vs
  , meanMatchCandidates = mmc
  , verbose=FALSE
)
```

`plotCorrelations` *plotCorrelations*

Description

Plot the correlation of imputed values between every combination of datasets for each variable.

Usage

```
plotCorrelations(
  miceObj,
  vars = names(miceObj$callParams$vars),
  factCorrMetric = "CramerV",
  numbCorrMetric = "pearson",
  ...
)
```

Arguments

`miceObj`	an object of class miceDefs, created by the miceRanger function.
`vars`	the variables you want to plot. Default is to plot all variables. Can be a vector of variable names, or one of 'allNumeric' or 'allCategorical'
`factCorrMetric`	The correlation metric for categorical variables. Can be one of: • `"CramerV"` Cramer's V correlation metric. • `"Chisq"` Chi Square test statistic. • `"TschuprowT"` Tschuprow's T correlation metric. • `"Phi"` (Binary Variables Only) Phi coefficient. • `"YuleY"` (Binary Variables Only) Yule's Y, also known as coefficient of colligation • `"YuleQ"` (Binary Variables Only) Yule's Q, related to Yule's Y by Q=2Y/(1+Y^2)
`numbCorrMetric`	The correlation metric for numeric variables. Can be one of: • `"pearson"` Pearson's Correlation Coefficient • `"spearman"` Spearman's Rank Correlation Coefficient • `"kendall"` Kendall's Rank Correlation Coefficient • `"Rsquared"` R-squared
`...`	Other arguments to pass to ggarrange()

Value

an object of class `ggarrange`.

Examples

```
data("sampleMiceDefs")
plotCorrelations(sampleMiceDefs)
```

plotDistributions *plotDistributions*

Description

Plots the distribution of the original data beside the imputed data.

Usage

```
plotDistributions(
  miceObj,
  vars = names(miceObj$callParams$vars),
  dotsize = 0.5,
  ...
)
```

Arguments

`miceObj`	an object of class miceDefs, created by the miceRanger function.
`vars`	the variables you want to plot. Default is to plot all variables. Can be a vector of variable names, or one of 'allNumeric' or 'allCategorical'
`dotsize`	Passed to `geom_dotplot()`. Depending on the number of graphs plotted, you may want to change the dot size for categorical variables.
`...`	additional parameters passed to `ggarrange()`.

Value

an object of class `ggarrange`.

Examples

```
data("sampleMiceDefs")
plotDistributions(sampleMiceDefs)
```

plotImputationVariance *plotImputationVariance*

Description

Plots the distribution of the difference between datasets of the imputed values. For categorical variables, the distribution of the number of distinct levels imputed for each sample is shown next to the distribution of unique draws from that variable in the nonmissing data, given that the draws were completely random. For numeric variables, the density of the standard deviation (between datasets) of imputations is plotted. The shaded area represents the samples that had a standard deviation lower than the total nonmissing standard deviation for the original data.

Usage

```
plotImputationVariance(
  miceObj,
  vars = names(miceObj$callParams$vars),
  monteCarloSimulations = 10000,
  ...
)
```

Arguments

`miceObj`	an object of class miceDefs, created by the miceRanger function.
`vars`	the variables you want to plot. Default is to plot all variables. Can be a vector of variable names, or one of 'allNumeric' or 'allCategorical'
`monteCarloSimulations`	The number of simulations to run to determine the distribution of unique categorical levels drawn if the draws were completely random.
`...`	additional parameters passed to `ggarrange()`.

Value

an object of class `ggarrange`.

Examples

```
data("sampleMiceDefs")
plotImputationVariance(
  sampleMiceDefs
  , monteCarloSimulations = 100
)
```

`plotModelError` *plotModelError*

Description

Plot the Out Of Bag model error for specified variables over all iterations.

Usage

```
plotModelError(
  miceObj,
  vars = names(miceObj$callParams$vars),
  pointSize = 1,
  ...
)
```

Arguments

`miceObj`	an object of class miceDefs, created by the miceRanger function.
`vars`	the variables you want to plot. Default is to plot all variables. Can be a vector of variable names, or one of 'allNumeric' or 'allCategorical'
`pointSize`	passed to `geom_point`, allows user to change dot size.
`...`	other arguments passed to `ggarrange()`

Value

an object of class `ggarrange`.

Examples

```
data("sampleMiceDefs")
plotModelError(sampleMiceDefs)
```

`plotVarConvergence` *plotVarConvergence*

Description

Plot the evolution of the dispersion and center of each variable. For numeric variables, the center is the mean, and the dispersion is the standard deviation. For categorical variables, the center is the mode, and the dispersion is the entropy of the distribution.

Usage

```
plotVarConvergence(miceObj, vars = names(miceObj$callParams$vars), ...)
```

Arguments

`miceObj`	an object of class `miceDefs`, created by the `miceRanger` function.
`vars`	the variables you want to plot. Default is to plot all variables. Can be a vector of variable names, or one of 'allNumeric' or 'allCategorical'
`...`	options passed to `ggarrange()`

Value

an object of class `ggarrange`.

Examples

```
data("sampleMiceDefs")
plotVarConvergence(sampleMiceDefs)
```

`plotVarImportance` *plotVarImportance*

Description

Plot the variable importance for each imputed variable. The values represent the variables on the top axis importance in imputing the variables on the left axis.

Usage

```
plotVarImportance(
  miceObj,
  display = c("Relative", "Absolute"),
  dataset = 1,
  ...
)
```

Arguments

`miceObj`	an object of class miceDefs, created by the miceRanger function.
`display`	How do you want to display variable importance? • "Relative" Scales the importance measure between 0-1 for each variable. • "Absolute" Displays the variable importance as is. May be highly skewed.
`dataset`	The dataset you want to plot the variable importance of.
`...`	Other arguments passed to `corrplot()`.

Examples

```
data("sampleMiceDefs")
plotVarImportance(sampleMiceDefs)
```

`print.miceDefs` *Print a* `miceDefs` *object*

Description

Print a `miceDefs` object

Usage

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

Arguments

`x`	Object of class `miceDefs`
`...`	required to use S3 method

Value

`NULL`

`sampleMiceDefs` *Sample miceDefs object built off of iris dataset. Included so examples don't run for too long.*

Description

Sample miceDefs object built off of iris dataset. Included so examples don't run for too long.

Usage

```
sampleMiceDefs
```

Format

A miceDefs object. See "'?miceRanger"' for details.

Source

set.seed(1991) data(iris) ampIris <- amputeData(iris,cols = c("Petal.Width","Species")) sampleMiceDefs <- miceRanger(ampIris ,m=3 ,maxiter=3 ,vars=c("Petal.Width","Species"))

Examples

```
## Not run:
 sampleMiceDefs

## End(Not run)
```

Index