Package 'miceFast'

August 20, 2019

Title Fast Imputations Using 'Rcpp' and 'Armadillo'

Version 0.5.1

Description Fast imputations under the object-oriented programming paradigm.
There was used quantitative models with a closed-
form solution. Thus package is based on linear algebra operations.
The biggest improvement in time performance could be achieve for a calculation where a grouping variable have to be used.
A single evaluation of a quantitative model for the multiple imputations is another major enhancement.
Moreover there are offered a few functions built to work with popular R packages such as 'data.table' or 'dplyr'.

Depends R (>= 3.4.0)

License GPL (>= 2)

URL https://github.com/Polkas/miceFast

BugReports https://github.com/Polkas/miceFast/issues

Encoding UTF-8

Imports methods, data.table, Rcpp (>= 0.12.12)

Suggests dplyr, knitr, rmarkdown, pacman, testthat, mice, broom, car, magrittr, ggplot2

VignetteBuilder knitr

LinkingTo Rcpp, RcppArmadillo

RcppModules miceFast,CorrData

SystemRequirements C++11

NeedsCompilation yes

LazyData true

RoxygenNote 6.1.0

Author Maciej Nasinski [aut, cre]

Maintainer Maciej Nasinski <nasinski.maciej@gmail.com>

Repository CRAN

Date/Publication 2019-08-19 22:50:02 UTC

R topics documented:

miceFast-package 2
air_miss 3
fill_NA 4
fill_NA_N 8
Rcpp_corrData-class 12
Rcpp_miceFast-class 13
VIF 14

Index **16**

`miceFast-package` *miceFast package for fast multiple imputations.*

Description

Fast imputations under the object-oriented programming paradigm. There was used quantitative models with a closed-form solution. Thus package is based on linear algebra operations. The biggest improvement in time performance could be achieve for a calculation where a grouping variable have to be used. A single evaluation of a quantitative model for the multiple imputations is another major enhancement. Moreover there are offered a few functions built to work with popular R packages such as 'data.table'.

Details

read vignette for additional information

Author(s)

Maciej Nasinski

References

.

See Also

.

Examples

```
  ## Not run:
.

## End(Not run)
```

`air_miss` *airquality dataset with additional variables.*

Description

airquality dataset with additional variables.

Usage

```
air_miss
```

Format

A data frame and data table with 154 observations on 11 variables.

Ozone numeric Ozone (ppb) - Mean ozone in parts per billion from 1300 to 1500 hours at Roosevelt Island

Solar.R numeric Solar R (lang) - Solar radiation in Langleys in the frequency band 4000–7700 Angstroms from 0800 to 1200 hours at Central Park

Wind numeric Wind (mph) - Average wind speed in miles per hour at 0700 and 1000 hours at LaGuardia Airport

Temp numeric Temperature (degrees F) - Maximum daily temperature in degrees Fahrenheit at La Guardia Airport.

Day numeric Day of month (1–31)

Intercept numeric a constant

index numeric id

weights numeric positive values weights

groups factor Month (1–12)

x_character character discrete version of Solar.R (5-levels)

Ozone_chac character discrete version of Ozone (7-levels)

Details

Daily readings of the following air quality values for May 1, 1973 (a Tuesday) to September 30, 1973.

Source

The data were obtained from the New York State Department of Conservation (ozone data) and the National Weather Service (meteorological data).

References

Chambers, J. M., Cleveland, W. S., Kleiner, B. and Tukey, P. A. (1983) Graphical Methods for Data Analysis. Belmont, CA: Wadsworth.

Examples

```
## Not run:
library(data.table)
data(airquality)
data = cbind(as.matrix(airquality[,-5]),Intercept=1,index=1:nrow(airquality),
            # a numeric vector - positive values
            weights = rnorm(nrow(airquality),1,0.01),
            # months as groups
            groups = airquality[,5])

# data.table
air_miss = data.table(data)
air_miss$groups = factor(air_miss$groups)

# Distribution of Ozone - close to log-normal
#hist(air_miss$Ozone)

# Additional vars
# Make a character variable to show package capabilities
air_miss$x_character = as.character(cut(air_miss$Solar.R,seq(0,350,70)))
# Discrete version of dependent variable
air_miss$Ozone_chac = as.character(cut(air_miss$Ozone, seq(0,160,20)))

## End(Not run)
```

`fill_NA` `fill_NA` *function for the imputations purpose.*

Description

Regular imputations to fill the missing data. Non missing independent variables are used to approximate a missing observations for a dependent variable. Quantitative models were built under Rcpp packages and the C++ library Armadillo.

Usage

```
fill_NA(x, model, posit_y, posit_x, w = NULL, logreg = FALSE)
```

Arguments

`x`	a numeric matrix or data.frame/data.table (factor/character/numeric) - variables
`model`	a character - posibble options ("lda","lm_pred","lm_bayes","lm_noise")
`posit_y`	an integer/character - a position/name of dependent variable
`posit_x`	an integer/character vector - positions/names of independent variables
`w`	a numeric vector - a weighting variable - only positive values, Default:NULL
`logreg`	a boolean - if dependent variable has log-normal distribution (numeric). If TRUE log-regression is evaluated and then returned exponential of results., Default: FALSE

Value

load imputations in a numeric/character/factor (similar to the input type) vector format

Note

There is assumed that users add the intercept by their own. The miceFast module provides the most efficient environment, the second recommended option is to use data.table and the numeric matrix data type. The lda model is assessed only if there are more than 15 complete observations and for the lms models if number of independent variables is smaller than number of observations.

See Also

fill_NA_N VIF

Examples

```
## Not run:
# install.packages('pacman')
pacman::p_load(miceFast,data.table,magrittr,dplyr)
### Data
# airquality dataset with additional variables
data(air_miss)

### Intro: data.table
# IMPUTATIONS
# Imputations with a grouping option (models are separately assessed for each group)
# taking into account provided weights
air_miss[,Solar_R_imp := fill_NA_N(x=.SD,
                                   model="lm_bayes",
                                   posit_y='Solar.R',
                                   posit_x=c('Wind','Temp','Intercept'),
                                   w=.SD[['weights']],
                                   times=100),by=.(groups)] %>%
# Imputations - discrete variable
  .[,x_character_imp := fill_NA(x=.SD,
                                model="lda",
                                posit_y='x_character',
                                posit_x=c('Wind','Temp','groups'))] %>%
# logreg was used because almost log-normal distribution of Ozone
# imputations around mean
.[,Ozone_imp1 := fill_NA(x=.SD,
                         model="lm_bayes",
                         posit_y='Ozone',
                         posit_x=c('Intercept'),
                         logreg=TRUE)] %>%
# imputations using positions - Intercept, Temp
.[,Ozone_imp2 := fill_NA(x=.SD,
                         model="lm_bayes",
                         posit_y=1,
                         posit_x=c(4,6),
                         logreg=TRUE)] %>%
# model with a factor independent variable
```

```
# multiple imputations (average of x30 imputations)
# with a factor independent variable, weights and logreg options
  .[,Ozone_imp3 := fill_NA_N(x=.SD,
                             model="lm_noise",
                             posit_y='Ozone',
                             posit_x=c('Intercept','x_character_imp','Wind','Temp'),
                             w=.SD[['weights']],
                             logreg=TRUE,
                             times=30)] %>%
  .[,Ozone_imp4 := fill_NA_N(x=.SD,
                             model="lm_bayes",
                             posit_y='Ozone',
                             posit_x=c('Intercept','x_character_imp','Wind','Temp'),
                             w=.SD[['weights']],
                             logreg=TRUE,
                             times=30)] %>%
  .[,Ozone_imp5 := fill_NA(x=.SD,
                           model="lm_pred",
                           posit_y='Ozone',
                           posit_x=c('Intercept','x_character_imp','Wind','Temp'),
                           w=.SD[['weights']],
                           logreg=TRUE),.(groups)] %>%

# Average of a few methods
  .[,Ozone_imp_mix := apply(.SD,1,mean),.SDcols=Ozone_imp1:Ozone_imp5] %>%

# Protecting against collinearity or low number of observations - across small groups
# Be carful when using a data.table grouping option
# because of lack of protection against collinearity or low number of observations.
# There could be used a tryCatch(fill_NA(...),error=function(e) return(...))

  .[,Ozone_chac_imp := tryCatch(fill_NA(x=.SD,
                                model="lda",
                                posit_y='Ozone_chac',
                                posit_x=c('Intercept',
                                          'Month',
                                          'Day',
                                          'Temp',
                                          'x_character_imp'),
                                w=.SD[['weights']]),
                               error=function(e) .SD[['Ozone_chac']]),.(groups)]

# Sample of results
air_miss[which(is.na(air_miss[,1]))[1:5],]

### Intro: dplyr
# IMPUTATIONS
air_miss = air_miss %>%
# Imputations with a grouping option (models are separately assessed for each group)
# taking into account provided weights
group_by(groups) %>%
do(mutate(.,Solar_R_imp = fill_NA(x=.,
                                  model="lm_pred",
```

```
                                posit_y='Solar.R',
                                posit_x=c('Wind','Temp','Intercept'),
                                w=.[['weights']]))) %>%
  ungroup() %>%
  # Imputations - discrete variable
  mutate(x_character_imp = fill_NA(x=.,
                                model="lda",
                                posit_y='x_character',
                                posit_x=c('Wind','Temp'))) %>%
  # logreg was used because almost log-normal distribution of Ozone
  # imputations around mean
  mutate(Ozone_imp1 = fill_NA(x=.,
                              model="lm_bayes",
                              posit_y='Ozone',
                              posit_x=c('Intercept'),
                              logreg=TRUE)) %>%
  # imputations using positions - Intercept, Temp
  mutate(Ozone_imp2 = fill_NA(x=.,
                              model="lm_bayes",
                              posit_y=1,
                              posit_x=c(4,6),
                              logreg=TRUE)) %>%
  # multiple imputations (average of x30 imputations)
  # with a factor independent variable, weights and logreg options
  mutate(Ozone_imp3 = fill_NA_N(x=.,
                                model="lm_noise",
                                posit_y='Ozone',
                                posit_x=c('Intercept','x_character_imp','Wind','Temp'),
                                w=.[['weights']],
                                logreg=TRUE,
                                times=30)) %>%
  mutate(Ozone_imp4 = fill_NA_N(x=.,
                                model="lm_bayes",
                                posit_y='Ozone',
                                posit_x=c('Intercept','x_character_imp','Wind','Temp'),
                                w=.[['weights']],
                                logreg=TRUE,
                                times=30)) %>%
  group_by(groups) %>%
  do(mutate(.,Ozone_imp5 = fill_NA(x=.,
                                model="lm_pred",
                                posit_y='Ozone',
                                posit_x=c('Intercept','x_character_imp','Wind','Temp'),
                                w=.[['weights']],
                                logreg=TRUE))) %>%
  ungroup() %>%
  # Average of a few methods
  mutate(Ozone_imp_mix = rowMeans(select(.,starts_with("Ozone_imp")))) %>%

  # Protecting against collinearity or low number of observations - across small groups
  # Be carful when using a data.table grouping option
  # because of lack of protection against collinearity or low number of observations.
  # There could be used a tryCatch(fill_NA(...),error=function(e) return(...))
```

```
group_by(groups) %>%
do(mutate(.,Ozone_chac_imp = tryCatch(fill_NA(x=.,
                                      model="lda",
                                      posit_y='Ozone_chac',
                                      posit_x=c('Intercept',
                                                'Month',
                                                'Day',
                                                'Temp',
                                                'x_character_imp'),
                                      w=.[['weights']]),
                                     error=function(e) .[['Ozone_chac']]))) %>%
ungroup()

# Sample of results
air_miss[which(is.na(air_miss[,1]))[1:5],]

## End(Not run)
```

`fill_NA_N` `fill_NA_N` *function for the multiple imputations purpose.*

Description

Multiple imputations to fill the missing data. Non missing independent variables are used to approximate a missing observations for a dependent variable. Quantitative models were built under Rcpp packages and the C++ library Armadillo.

Usage

```
fill_NA_N(x, model, posit_y, posit_x, w = NULL, logreg = FALSE,
  times = 10)
```

Arguments

`x`	a numeric matrix or data.frame/data.table (factor/character/numeric) - variables
`model`	a character - posibble options ("lm_bayes","lm_noise")
`posit_y`	an integer/character - a position/name of dependent variable
`posit_x`	an integer/character vector - positions/names of independent variables
`w`	a numeric vector - a weighting variable - only positive values, Default: NULL
`logreg`	a boolean - if dependent variable has log-normal distribution (numeric). If TRUE log-regression is evaluated and then returned exponential of results., Default: FALSE
`times`	an integer - a number of multiple imputations, Default:10

Value

load imputations in a numeric/character/factor (similar to the input type) vector format

Note

There is assumed that users add the intercept by their own. The miceFast module provides the most efficient environment, the second recommended option is to use data.table and the numeric matrix data type. The lda model is assessed only if there are more than 15 complete observations and for the lms models if number of variables is smaller than number of observations.

See Also

fill_NA VIF

Examples

```
## Not run:
# install.packages('pacman')
pacman::p_load(miceFast,data.table,magrittr,dplyr)
### Data
# airquality dataset with additional variables
data(air_miss)

### Intro: data.table
# IMPUTATIONS
# Imputations with a grouping option (models are separately assessed for each group)
# taking into account provided weights
air_miss[,Solar_R_imp := fill_NA_N(x=.SD,
                                   model="lm_bayes",
                                   posit_y='Solar.R',
                                   posit_x=c('Wind','Temp','Intercept'),
                                   w=.SD[['weights']],
                                   times=100),by=.(groups)] %>%
# Imputations - discrete variable
  .[,x_character_imp := fill_NA(x=.SD,
                                model="lda",
                                posit_y='x_character',
                                posit_x=c('Wind','Temp','groups'))] %>%
# logreg was used because almost log-normal distribution of Ozone
# imputations around mean
.[,Ozone_imp1 := fill_NA(x=.SD,
                         model="lm_bayes",
                         posit_y='Ozone',
                         posit_x=c('Intercept'),
                         logreg=TRUE)] %>%
# imputations using positions - Intercept, Temp
.[,Ozone_imp2 := fill_NA(x=.SD,
                         model="lm_bayes",
                         posit_y=1,
                         posit_x=c(4,6),
                         logreg=TRUE)] %>%
# model with a factor independent variable
# multiple imputations (average of x30 imputations)
# with a factor independent variable, weights and logreg options
  .[,Ozone_imp3 := fill_NA_N(x=.SD,
                             model="lm_noise",
```

```
                              posit_y='Ozone',
                              posit_x=c('Intercept','x_character_imp','Wind','Temp'),
                              w=.SD[['weights']],
                              logreg=TRUE,
                              times=30)] %>%
  .[,Ozone_imp4 := fill_NA_N(x=.SD,
                              model="lm_bayes",
                              posit_y='Ozone',
                              posit_x=c('Intercept','x_character_imp','Wind','Temp'),
                              w=.SD[['weights']],
                              logreg=TRUE,
                              times=30)] %>%
  .[,Ozone_imp5 := fill_NA(x=.SD,
                           model="lm_pred",
                           posit_y='Ozone',
                           posit_x=c('Intercept','x_character_imp','Wind','Temp'),
                           w=.SD[['weights']],
                           logreg=TRUE),.(groups)] %>%

# Average of a few methods
  .[,Ozone_imp_mix := apply(.SD,1,mean),.SDcols=Ozone_imp1:Ozone_imp5] %>%

# Protecting against collinearity or low number of observations - across small groups
# Be carful when using a data.table grouping option
# because of lack of protection against collinearity or low number of observations.
# There could be used a tryCatch(fill_NA(...),error=function(e) return(...))

  .[,Ozone_chac_imp := tryCatch(fill_NA(x=.SD,
                                model="lda",
                                posit_y='Ozone_chac',
                                posit_x=c('Intercept',
                                          'Month',
                                          'Day',
                                          'Temp',
                                          'x_character_imp'),
                                w=.SD[['weights']]),
                               error=function(e) .SD[['Ozone_chac']]),.(groups)]

# Sample of results
air_miss[which(is.na(air_miss[,1]))[1:5],]

### Intro: dplyr
# IMPUTATIONS
air_miss = air_miss %>%
# Imputations with a grouping option (models are separately assessed for each group)
# taking into account provided weights
group_by(groups) %>%
do(mutate(.,Solar_R_imp = fill_NA(x=.,
                                  model="lm_pred",
                                  posit_y='Solar.R',
                                  posit_x=c('Wind','Temp','Intercept'),
                                  w=.[['weights']]))) %>%
ungroup() %>%
```

```
# Imputations - discrete variable
mutate(x_character_imp = fill_NA(x=.,
                                 model="lda",
                                 posit_y='x_character',
                                 posit_x=c('Wind','Temp'))) %>%
# logreg was used because almost log-normal distribution of Ozone
# imputations around mean
mutate(Ozone_imp1 = fill_NA(x=.,
                            model="lm_bayes",
                            posit_y='Ozone',
                            posit_x=c('Intercept'),
                            logreg=TRUE)) %>%
# imputations using positions - Intercept, Temp
mutate(Ozone_imp2 = fill_NA(x=.,
                            model="lm_bayes",
                            posit_y=1,
                            posit_x=c(4,6),
                            logreg=TRUE)) %>%
# multiple imputations (average of x30 imputations)
# with a factor independent variable, weights and logreg options
mutate(Ozone_imp3 = fill_NA_N(x=.,
                              model="lm_noise",
                              posit_y='Ozone',
                              posit_x=c('Intercept','x_character_imp','Wind','Temp'),
                              w=.[['weights']],
                              logreg=TRUE,
                              times=30)) %>%
mutate(Ozone_imp4 = fill_NA_N(x=.,
                              model="lm_bayes",
                              posit_y='Ozone',
                              posit_x=c('Intercept','x_character_imp','Wind','Temp'),
                              w=.[['weights']],
                              logreg=TRUE,
                              times=30)) %>%
group_by(groups) %>%
do(mutate(.,Ozone_imp5 = fill_NA(x=.,
                                 model="lm_pred",
                                 posit_y='Ozone',
                                 posit_x=c('Intercept','x_character_imp','Wind','Temp'),
                                 w=.[['weights']],
                                 logreg=TRUE))) %>%
ungroup() %>%
# Average of a few methods
mutate(Ozone_imp_mix = rowMeans(select(.,starts_with("Ozone_imp")))) %>%

# Protecting against collinearity or low number of observations - across small groups
# Be carful when using a data.table grouping option
# because of lack of protection against collinearity or low number of observations.
# There could be used a tryCatch(fill_NA(...),error=function(e) return(...))
group_by(groups) %>%
do(mutate(.,Ozone_chac_imp = tryCatch(fill_NA(x=.,
                                      model="lda",
                                      posit_y='Ozone_chac',
```

```
                                        posit_x=c('Intercept',
                                                  'Month',
                                                  'Day',
                                                  'Temp',
                                                  'x_character_imp'),
                                        w=.[['weights']]),
                                       error=function(e) .[['Ozone_chac']]))) %>%
  ungroup()

  # Sample of results
  air_miss[which(is.na(air_miss[,1]))[1:5],]


  ## End(Not run)
```

Rcpp_corrData-class *Class* "Rcpp_corrData"

Description

This C++ class could be used to build a corrData object by invoking `new(corrData,...)` function.

Extends

Class "C++Object", directly.

All reference classes extend and inherit methods from "envRefClass".

Methods

`initialize(...)`: ~~

`finalize()`: ~~

`fill(...)`: generating data

Note

This is only frame for building C++ object which could be used to implement certain methods. Check the vignette for more details of implementing methods.

References

See the documentation for RcppArmadillo and Rcpp for more details of how this class was built.

Examples

```
#showClass("Rcpp_corrData")
show(corrData)
```

Rcpp_miceFast-class *Class* "Rcpp_miceFast"

Description

This C++ class could be used to build a miceFast objects by invoking `new(miceFast)` function.

Extends

Class "C++Object", directly.

All reference classes extend and inherit methods from "envRefClass".

Methods

`set_data(...)`: providing data by a reference - a numeric matrix

`set_g(...)`: providing a grouping variable by a reference - a numeric vector - positive values

`set_w(...)`: providing a weightinh variable by a reference - a numeric vector - positive values

`get_data(...)`: retrieving the data

`get_w(...)`: retrieving the weighting variable

`get_g(...)`: retireiving the grouping variable

`get_index(...)`: getting the index

`impute(...)`: impute data under characterstics from the object like a optional grouping or weighting variable

`impute_N(...)`: multiple imputations - impute data under characterstics from the object like a optional grouping or weighting variable

`update_var(...)`: permanently update the variable at the object and data. Use it only if you are sure about model parameters

`get_models(...)`: get possible quantitative models for a certain type of dependent variable

`get_model(...)`: get a recommended quantitative model for a certain type of dependent variable

`which_updated(...)`: which variables at the object was modified by update_var

`sort_byg(...)`: sort data by the grouping variable

`is_sorted_byg(...)`: check if data is sorted by the grouping variable

`vifs(...)`: Variance inflation factors (VIF) - helps to check when the predictor variables are not linearly related

`initialize(...)`: ...

`finalize()`: ...

Note

This is only frame for building C++ object which could be used to implement certain methods. Check the vignette for more details of implementing these methods.

Vigniette: https://CRAN.R-project.org/package=miceFast

References

See the documentation for RcppArmadillo and Rcpp for more details of how this class was built.

Examples

```
#showClass("Rcpp_miceFast")
show(miceFast)
new(miceFast)
```

VIF VIF *function for assessing VIF.*

Description

VIF measure how much the variance of the estimated regression coefficients are inflated. It helps to identify when the predictor variables are linearly related. You have to decide which variable should be delete. Values higher than 10 signal a potential collinearity problem.

Usage

```
VIF(x, posit_y, posit_x, correct = FALSE)

## S3 method for class 'data.frame'
VIF(x, posit_y, posit_x, correct = FALSE)

## S3 method for class 'matrix'
VIF(x, posit_y, posit_x, correct = FALSE)
```

Arguments

x	a numeric matrix or data.frame/data.table (factor/character/numeric) - variables
posit_y	an integer/character - a position/name of dependent variable
posit_x	an integer/character vector - positions/names of independent variables
correct	a boolean - basic or corrected - Default: FALSE

Value

load a numeric vector with VIF for all variables provided by posit_x

Methods (by class)

- data.frame:
- matrix:

Note

vif_corrected = vif_basic^(1/(2*df))

See Also

fill_NA fill_NA_N

Examples

```
## Not run:
library(miceFast)
library(data.table)

airquality2 = airquality
airquality2$Temp2 = airquality2$Temp**2
#install.packages("car")
#car::vif(lm(Ozone ~ ., data=airquality2))

data_DT = data.table(airquality2)
data_DT[,.(vifs=VIF(x=.SD,
                    posit_y='Ozone',
                    posit_x=c('Solar.R','Wind','Temp','Month','Day','Temp2'),
                    correct=FALSE))][['vifs']]

data_DT[,.(vifs=VIF(x=.SD,
                    posit_y=1,
                    posit_x=c(2,3,4,5,6,7),
                    correct=TRUE))][['vifs']]


## End(Not run)
```

Index

∗Topic **classes**
    Rcpp_corrData-class, 12
    Rcpp_miceFast-class, 13
∗Topic **datasets**
    air_miss, 3
∗Topic **package**
    miceFast-package, 2

air_miss, 3

C++Object, *12, 13*
corrData *(*Rcpp_corrData-class*)*, 12

envRefClass, *12, 13*

fill_NA, 4, *9*, *15*
fill_NA_N, *5*, 8, *15*

miceFast *(*Rcpp_miceFast-class*)*, 13
miceFast-package, 2

Rcpp_corrData-class, 12
Rcpp_miceFast-class, 13

VIF, *5*, *9*, 14