Package 'miceFast'

May 6, 2018

Title Fast Imputations Using 'Rcpp' and 'Armadillo'

Version 0.2.3

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'.

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, Rcpp (>= 0.12.12)

Suggests data.table, knitr, rmarkdown, pacman, testthat, mice, dplyr,
broom, car, magrittr

VignetteBuilder knitr

LinkingTo Rcpp, RcppArmadillo

RcppModules miceFast,CorrData

SystemRequirements C++11

NeedsCompilation yes

RoxygenNote 6.0.1

Author Maciej Nasinski [aut, cre]

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

Repository CRAN

Date/Publication 2018-05-06 20:19:04 UTC

R topics documented:

miceFast-package 2
fill_NA 3
fill_NA_N 4
Rcpp_corrData-class 6
Rcpp_miceFast-class 7
VIF 8

Index **10**

`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.

Details

read vignette for additional information

Author(s)

Maciej Nasinski

References

.

See Also

.

Examples

```
  ## Not run:
.

## 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 = 0L)
```

Arguments

x	a numeric matrix - a numeric matrix with variables
model	a character - posibble options ("lda","lm_pred","lm_bayes","lm_noise")
posit_y	an integer - a position of dependent variable
posit_x	an integer vector - positions of independent variables
w	a numeric vector - a weighting variable - only positive values

Value

load variable at position y with additional imputations in a numeric vector format

Note

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:
library(miceFast)
library(data.table)
library(magrittr)

data = cbind(as.matrix(airquality[,-5]),intercept=1,index=1:nrow(airquality),
             # a numeric vector - positive values
             weights = round(rgamma(nrow(airquality),3,3),1),
             # as.numeric is needed only for OOP miceFast - see on next pages
             groups = airquality[,5])
data_DT = data.table(data)
```

```
# simple mean imputation - intercept at position 6
data_DT[,Ozone_imp:=fill_NA(x=as.matrix(.SD),
                            model="lm_pred",
                            posit_y=1,
                            posit_x=c(6),w=.SD[['weights']]),by=.(groups)] %>%
# avg of 10 multiple imputations - last posit_x equal to 9 not 10
# because the groups variable is not included in .SD
.[,Solar_R_imp:=fill_NA_N(as.matrix(.SD),
                          model="lm_bayes",
                          posit_y=2,
                          posit_x=c(3,4,5,6,9),w=.SD[['weights']],times=10),by=.(groups)]

head(data_DT,10)

######################
#OR using OOP miceFast
######################

data = cbind(as.matrix(airquality[,-5]),intercept=1,index=1:nrow(airquality))
weights = rgamma(nrow(data),3,3) # a numeric vector - positive values
#a numeric vector not integers - positive values - sorted increasingly
groups = as.numeric(airquality[,5])
#a numeric vector not integers - positive values - not sorted
#groups = as.numeric(sample(1:8,nrow(data),replace=T))

model = new(miceFast)
model$set_data(data) # providing data by a reference
model$set_w(weights) # providing by a reference
model$set_g(groups)  # providing by a reference

#impute adapt to provided parmaters like w or g
#Simple mean - permanent imputation at the object and data
#variable will be replaced by imputations
model$update_var(1,model$impute("lm_pred",1,c(6))$imputations)

model$update_var(2,model$impute_N("lm_bayes",2,c(1,3,4,5,6),10)$imputations)

#Printing data and retrieving an old order if data was sorted by the grouping variable
head(cbind(model$get_data(),model$get_g(),model$get_w())[order(model$get_index()),],3)
#the same
head(cbind(data,groups,weights)[order(model$get_index()),],3)


## 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 = 0L, times = 10L)
```

Arguments

x	a numeric matrix - a numeric matrix with variables
model	a character - posibble options ("lm_bayes","lm_noise")
posit_y	an integer - a position of dependent variable
posit_x	an integer vector - positions of independent variables
w	a numeric vector - a weighting variable - only positive values
times	an integer - a number of multiple imputations - default 10

Value

load variable at position y with additional average of N imputations in a numeric vector format

Note

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:
library(miceFast)
library(data.table)
library(magrittr)

data = cbind(as.matrix(airquality[,-5]),intercept=1,index=1:nrow(airquality),
             # a numeric vector - positive values
             weights = round(rgamma(nrow(airquality),3,3),1),
             # as.numeric is needed only for OOP miceFast - see on next pages
             groups = airquality[,5])
data_DT = data.table(data)

# simple mean imputation - intercept at position 6
data_DT[,Ozone_imp:=fill_NA(x=as.matrix(.SD),
                            model="lm_pred",
                            posit_y=1,
```

```
                            posit_x=c(6),w=.SD[['weights']]),by=.(groups)] %>%
# avg of 10 multiple imputations - last posit_x equal to 9 not 10
# because the groups variable is not included in .SD
.[,Solar_R_imp:=fill_NA_N(as.matrix(.SD),
                          model="lm_bayes",
                          posit_y=2,
                          posit_x=c(3,4,5,6,9),w=.SD[['weights']],times=10),by=.(groups)]

head(data_DT,10)

######################
#OR using OOP miceFast
######################

data = cbind(as.matrix(airquality[,-5]),intercept=1,index=1:nrow(airquality))
weights = rgamma(nrow(data),3,3) # a numeric vector - positive values
#a numeric vector not integers - positive values - sorted increasingly
groups = as.numeric(airquality[,5])
#a numeric vector not integers - positive values - not sorted
#groups = as.numeric(sample(1:8,nrow(data),replace=T))

model = new(miceFast)
model$set_data(data) # providing data by a reference
model$set_w(weights) # providing by a reference
model$set_g(groups)  # providing by a reference

#impute adapt to provided parmaters like w or g
#Simple mean - permanent imputation at the object and data
#variable will be replaced by imputations
model$update_var(1,model$impute("lm_pred",1,c(6))$imputations)

model$update_var(2,model$impute_N("lm_bayes",2,c(1,3,4,5,6),10)$imputations)

#Printing data and retrieving an old order if data was sorted by the grouping variable
head(cbind(model$get_data(),model$get_g(),model$get_w())[order(model$get_index()),],3)
#the same
head(cbind(data,groups,weights)[order(model$get_index()),],3)



## 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)
```

Arguments

x	a numeric matrix - a numeric matrix with variables
posit_y	an integer - a position of dependent variable
posit_x	an integer vector - positions of independent variables

Value

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

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)
# VIF for variables at 1,3,4 positions - you include a y position to consider its NA values
data_DT[,.(vifs=VIF(x=as.matrix(.SD),
                    posit_y=1,
                    posit_x=c(2,3,4,5,6,7)))]

######################
#OR using OOP miceFast
######################

airquality2_mat = as.matrix(airquality2)
model = new(miceFast)
model$set_data(airquality2_mat)

as.vector(model$vifs(1,c(2,3,4,5,6,7)))


## End(Not run)
```

Index

∗Topic **classes**
    Rcpp_corrData-class, 6
    Rcpp_miceFast-class, 7
∗Topic **package**
    miceFast-package, 2

C++Object, *7*
corrData *(*Rcpp_corrData-class*)*, 6

envRefClass, *7*

fill_NA, 3, *5*, *9*
fill_NA_N, *3*, 4, *9*

miceFast *(*Rcpp_miceFast-class*)*, 7
miceFast-package, 2

Rcpp_corrData-class, 6
Rcpp_miceFast-class, 7

VIF, *3*, *5*, 8