miceFast - Introduction

Maciej Nasinski

2018-04-16

Loading the package and setting a seed:

```
library(miceFast)
set.seed(123456)
```

Performance

miceFast was compared[1] with the mice package. For grouping option there was used a basic R looping and the popular dplyr package. Summing up, miceFast offer a relevant boost of calculations for:

- Linear Discriminant Analysis around **(x10)**
- where a grouping variable have to be used **(around x50 depending on data dimensions and number of groups and even more than x1000)** because data is sorted by grouping variable*
- multiple imputatiotions is faster around **x(a number of multiple imputatition)** because the core of a model is evaluated only ones.

If you are interested about the procedure of testing performance check performance_validity.R file at extdata folder.

```
system.file("extdata","performance_validity.R",package = "miceFast")
```

Additinal plots for simulations with certain parmaeters (but feel free to change them) are located:

```
system.file("extdata","images",package = "miceFast")
```

Motivations

Missing data is a common problem. The easiest solution is to delete observations for which a certain variable is missing. However this will sometimes deteriorate quality of a project. Another solution will be to use methods such as multiple/regular imputations to fill the missing data. Non missing independent variables could be used to approximate a missing observations for a dependent variable. R or Python language are user-friendly for data manipulation but parallely brings slower computations. Languages such as C++ gives an opportunity to boost our applications or projects.

The presented miceFast package was built under Rcpp packages and the C++ library Armadillo. The Rcpp package offers functionality of exporting full C++ capabilities to the R environment. More precisely miceFast and corrData are offered. The first module offers capabilities of imputations models with a closed-form solution. Thus package is based on linear algebra operations. The main upgrade is possibility of including a grouping and/or weighting (only for linear models) variable and functions enhancement by C++ capabilities. The second module was made for purpose of presenting the miceFast usage and performance. It provides functionality of generating correlated data with a discrete, binomial or continuous dependent variable and continuous independent variables.

[1] Environment: MRO 3.4.1 Intel MKL - i7 6700HQ and 24GB DDR4 2133. MRO (Microsoft R Open) provide to R a sophisticated library for linear algebra operations so remember about that when reading a performance comparision.

Genereting data with the corrData Module

Available constructors:

`new(corrData,nr_cat,n_obs,means,cor_matrix)`

`new(corrData,n_obs,means,cor_matrix)`

where:

- `nr_cat` : number of categories for discrete dependent variable
- `n_obs` : number of observations
- `means`: center independent variables
- `cor_mat` : positive defined correlation matrix

relevant class methods:

- `fill("type")` : generating data

`type`:character - possible options ("contin","binom","discrete")

Imputing data with the miceFast Module:

Available constructors:

`new(miceFast)`

relevant class methods:

- `set_data(x)` - providing the data by a reference - a numeric matrix
- `get_data()` - retrieving the data
- `set_g(g)` - providing the grouping variable by a reference - a numeric vector - positive values
- `get_g()` - retrieving the grouping variable
- `set_w(w)` - providing the weighting variable by a reference - a numeric vector - positive values
- `get_w()` - retrieving the weighting variable
- `get_index()` - getting the index
- `impute("model",posit_y,posit_x)` - impute data under characterstics form object like a optional grouping or weighting variable
- `impute_N("model",posit_y,posit_x,times)` - multiple imputations - impute data under characterstics form object like a optional grouping or weighting variable - works for ("lm_bayes","lm_noise")
- `update_var(posit_y,imputations)` - permanently update variable at the object and data. WARNING, use it only if you are sure about model parameters.
- `get_models()` - possible quantitative models for a certain type of dependent variable
- `get_model()` - a recommended quantitative model for a certain type of dependent variable
- `which_updated()` - which variables were modified by update_var at the object (and data)
- `sort_byg()` - sort data by the grouping variable
- `is_sorted_byg()` - is data sorted by the grouping variable
- `vifs(posit_y,posit_x)` - Variance inflation factors (VIF)
  `x` : numeric matrix - variables
  `g` : numeric vector - you could build it form several discrete variables
  `w`: numeric vector with positive values - weights for weighted linear regressions
  `model`: character - posibble options ("lda","lm_pred","lm_bayes","lm_noise")
  `posit_y`: integer - position of dependent variable
  `posit_x`: integer vector - positions of independent variables
  `imputations` : numeric vector - imputations
  `times` : integer - number of multiple imputations

For a simple mean imputations add intercept to data and use "lm_pred"

Imputations

miceFast module usage:

```
#install.packages("mice")
data = cbind(as.matrix(mice::nhanes),intercept=1,index=1:nrow(mice::nhanes))
model = new(miceFast)
model$set_data(data) #providing data by a reference

model$update_var(2,model$impute("lm_pred",2,5)$imputations)
#OR not recommended
#data[,2] = model$impute("lm_pred",2,5)$imputations
#model$set_data(data) #Updating the object

model$update_var(3,model$impute("lda",3,c(1,2))$imputations)

#Old slow syntax
#model$update_var(4,rowMeans(sapply(1:10,function(x)
#  model$impute("lm_bayes",4,c(1,2,3))$imputations)))
model$update_var(4,model$impute_N("lm_bayes",4,c(1,2,3),10)$imputations)

#When working with 'Big Data'
#it is recommended to occasionally manually invoke a garbage collector `gc()`

# Be careful with `update_var` because of the permanent update at the object and data
# That is why `update_var` could be used only ones for a certain column
# check which variables was updated - inside the object
model$which_updated()
```

```
## [1] 2 3 4
```

```
head(model$get_data(),4)
```

```
##      [,1]    [,2] [,3]     [,4] [,5] [,6]
## [1,]    1 26.5625    1 161.2406    1    1
## [2,]    2 22.7000    1 187.0000    1    2
## [3,]    1 26.5625    1 187.0000    1    3
## [4,]    3 26.5625    2 240.9674    1    4
```

```
head(data,4)
```

```
##   age     bmi hyp      chl intercept index
## 1   1 26.5625   1 161.2406         1     1
## 2   2 22.7000   1 187.0000         1     2
## 3   1 26.5625   1 187.0000         1     3
## 4   3 26.5625   2 240.9674         1     4
```

```
head(mice::nhanes,4)
```

```
##   age  bmi hyp chl
## 1   1   NA  NA  NA
## 2   2 22.7   1 187
## 3   1   NA   1 187
## 4   3   NA  NA  NA
```

```
rm(model)
```

Model with additional parameters: - data sorted by the grouping variable

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

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
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
head(cbind(model$get_data(),model$get_g(),model$get_w())[order(model$get_index()),],4)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]      [,9]
## [1,]   41  190  7.4   67    1    1    1    5 0.8042191
## [2,]   36  118  8.0   72    2    1    2    5 1.5597434
## [3,]   12  149 12.6   74    3    1    3    5 0.4004544
## [4,]   18  313 11.5   62    4    1    4    5 0.4909557
```

```
head(airquality,4)
```

```
##   Ozone Solar.R Wind Temp Month Day
## 1    41     190  7.4   67     5   1
## 2    36     118  8.0   72     5   2
## 3    12     149 12.6   74     5   3
## 4    18     313 11.5   62     5   4
```

```
head(cbind(model$get_data(),model$get_g(),model$get_w()),4)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]      [,9]
## [1,]   41  190  7.4   67    1    1    1    5 0.8042191
## [2,]   36  118  8.0   72    2    1    2    5 1.5597434
## [3,]   12  149 12.6   74    3    1    3    5 0.4004544
## [4,]   18  313 11.5   62    4    1    4    5 0.4909557
```

```
head(cbind(data,groups,weights),4)
```

```
##      Ozone Solar.R Wind Temp Day intercept index groups   weights
## [1,]    41     190  7.4   67   1         1     1      5 0.8042191
## [2,]    36     118  8.0   72   2         1     2      5 1.5597434
## [3,]    12     149 12.6   74   3         1     3      5 0.4004544
## [4,]    18     313 11.5   62   4         1     4      5 0.4909557
```

```
rm(model)
```

Model with additional parameters: - data not sorted by the grouping variable

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

model = new(miceFast)
model$set_data(data) # providing by a reference
model$set_w(weights) # providing by a reference
model$set_g(groups)  # providing by a reference
#impute adapt to provided parmaters like w or g
#Warning - if data is not sorted increasingly by the g then it would be done automatically
#during a first imputation
#Simple mean - permanent imputation at the object and data
model$update_var(1,model$impute("lm_pred",1,6)$imputations)
```

```
## Warning in model$impute("lm_pred", 1, 6):
##  Data was sorted by the grouping variable - use `get_index()` to retrieve an order
```

```
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
head(cbind(model$get_data(),model$get_g(),model$get_w())[order(model$get_index()),],4)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]     [,9]
## [1,]   41  190  7.4   67    1    1    1    3 1.670560
## [2,]   36  118  8.0   72    2    1    2    2 1.041644
## [3,]   12  149 12.6   74    3    1    3    2 1.680337
## [4,]   18  313 11.5   62    4    1    4    1 0.200267
```

```
head(airquality,4)
```

```
##   Ozone Solar.R Wind Temp Month Day
## 1    41     190  7.4   67     5   1
## 2    36     118  8.0   72     5   2
## 3    12     149 12.6   74     5   3
## 4    18     313 11.5   62     5   4
```

```
head(cbind(model$get_data(),model$get_g(),model$get_w()),4) #is ordered by g
```

```
##      [,1]     [,2] [,3] [,4] [,5] [,6] [,7] [,8]      [,9]
## [1,]   18 313.0000 11.5   62    4    1    4    1 0.2002670
## [2,]   28 246.3366 14.9   66    6    1    6    1 0.6724756
## [3,]   19  99.0000 13.8   59    8    1    8    1 0.9913337
## [4,]    8  19.0000 20.1   61    9    1    9    1 0.3370238
```

```
head(cbind(data,groups,weights),4) #is sorted by g cause we provide data by a reference
```

```
##      Ozone  Solar.R Wind Temp Day intercept index groups   weights
## [1,]    18 313.0000 11.5   62   4         1     4      1 0.2002670
## [2,]    28 246.3366 14.9   66   6         1     6      1 0.6724756
## [3,]    19  99.0000 13.8   59   8         1     8      1 0.9913337
## [4,]     8  19.0000 20.1   61   9         1     9      1 0.3370238
```

```
rm(model)
```

Tips

matrix from data.frame

Remeber that a matrix could be build only under a one data type so factor/character variables have to be melted. Sb could use `model.matrix` to get numeric matrix from a `data.frame`:

```
#str(mtcars)
mtcars$cyl= factor(mtcars$cyl)
mtcars$gear= factor(mtcars$gear)
mtcars_mat = model.matrix(~.,mtcars)
#str(mtcars_mat)
```

Variance inflation factors (VIF)

VIF measure how much the variance of the estimated regression coefficients are inflated. It helps to indentify 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.

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

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)))
```

```
## [1]   1.176572   1.340626 227.509393   1.347135   1.011108 223.460971
```

Bibliography

URL: http://dirk.eddelbuettel.com/code/rcpp/Rcpp-modules.pdf
Title: Exposing C++ functions and classes with Rcpp modules Dirk Eddelbuettel and Romain François
Author: http://dirk.eddelbuettel.com and https://romain.rbind.io/
Date: March 8, 2018

URL: http://dirk.eddelbuettel.com/papers/RcppArmadillo-intro.pdf
Title: RcppArmadillo: Easily Extending R with High-Performance C++ Code
Author: Dirk Eddelbuettel and Conrad Sanderson
Date: July 1, 2012

URL: http://www.stefvanbuuren.nl/publications/MICE%20in%20R%20-%20Draft.pdf
Title: MICE: Multivariate Imputation by Chained Equations in R
Author: Stef van Buuren
Date: 2013

URL: http://dirk.eddelbuettel.com/code/rcpp/Rcpp-introduction.pdf
Title: Extending R with C++: A Brief Introduction to Rcpp
Author: Dirk Eddelbuettel and James Joseph Balamuta
Date: March 8, 2018

URL: http://courses.cs.tamu.edu/rgutier/csce666_f13/
Title: CSCE 666: Pattern Analysis
Author: Ricardo Gutierrez-Osuna
Date: Fall 2013