miceFast - Introduction

Maciej Nasinski

2018-03-18

Loading the package and setting a seed:

```r
library(miceFast)
set.seed(1234)
```

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 multiple 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.

Performance

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. The biggest improvement in time performance could be achieved for a calculation where a grouping variable have to be used (around x50 depending on data dimensions and number of groups and even more than x1000). Another performance boost could be achieved for Linear Discriminant Analysis (x10).

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

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

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

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

miceFast was compared 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 LDA and all implemented models with a grouping variable.

Modules

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
- `get_data()` - retrieving the data
- `set_g(g)` - providing the grouping variable
- `get_g()` - retrieving the grouping variable
- `set_w(w)` - providing the weighting variable
- `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
- `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 `x` : numeric matrix - variables
  `g` : vector of integers for grouping variable - 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

For a simple mean add intercept to data `rep(1,nrow(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))$imputations)
model$update_var(4,rowMeans(sapply(1:10,function(x)
  model$impute("lm_bayes",4,c(1,2,3))$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())
```

```
##      [,1]    [,2] [,3]     [,4] [,5] [,6]
## [1,]    1 26.5625    1 144.8093    1    1
## [2,]    2 22.7000    1 187.0000    1    2
## [3,]    1 26.5625    1 187.0000    1    3
## [4,]    3 26.5625    2 225.2138    1    4
## [5,]    1 20.4000    1 113.0000    1    5
## [6,]    3 26.5625    2 184.0000    1    6
```

```
rm(model)
```

```
####################################
###Model with additional parameters
####################################

data = cbind(as.matrix(airquality[,-5]),intercept=1,index=1:nrow(airquality))
weights = rgamma(nrow(data),3,3) # positive numeric values
#groups = airquality[,5] # vector of positive integers
groups = sample(1:4,nrow(data),replace=T) # vector of positive integers

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

#if data is not sorted increasingly by g then it would be automatically done
#during a first imputation

#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)
```

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

```
model$update_var(2,rowMeans(sapply(1:10,function(x)
  model$impute("lm_bayes",2,c(1,3,4,5,6))$imputations))
  )

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

```
##          [,1]          [,2] [,3] [,4] [,5] [,6] [,7] [,8]      [,9]
## [1,] 41.00000  1.900000e+02  7.4   67    1    1    1    3 0.8676524
## [2,] 36.00000  1.180000e+02  8.0   72    2    1    2    4 1.4157502
## [3,] 12.00000  1.490000e+02 12.6   74    3    1    3    4 0.7762213
## [4,] 18.00000  3.130000e+02 11.5   62    4    1    4    1 0.8430150
## [5,] 43.49289  2.612966e+02 14.3   56    5    1    5    3 1.2464766
## [6,] 28.00000 1.913833e-314 14.9   66    6    1    6    4 0.5272105
```

```
rm(model)
```