Atelier mice

Vincent Audigier

16 Novembre 2021

Packages R

Installation des packages requis (inutile si déjà installés)

```
install.packages(c("micemd","FactoMineR","parallel","DescTools",
                   "VIM","mice", "mvtnorm", "funModeling","randomForest"))
```

Chargement des packages (indispensable)

```
library(mice)
library(micemd)
library(FactoMineR)
library(parallel)
library(DescTools)
library(VIM)
library(mvtnorm)
library(funModeling)
```

Importation des données

```
load("diabetes.Rdata")
```

Une description du jeu de données est disponible ici : https://github.com/niharikagulati/diabetesprediction

Analyse exploratoire

1. Explorer le lien entre les variables explicatives d'une part et entre la variable réponse et les variables explicatives d'autre part.

```
dim(diabetes)
summary(diabetes)

pairs(diabetes[, 1:8], cex=.2)
matcor <- cor(diabetes[,1:8], use = "pairwise.complete.obs")
PlotCorr(matcor)

boxplot(diabetes[,"Age"]~diabetes$Outcome)

par(mfrow = c(3, 3))
sapply(colnames(diabetes)[1:8],
```

```
       FUN = function(xx, diabetes){
         boxplot(diabetes[,xx]~diabetes$Outcome,
                 main = xx,
                 ylab = xx,
                 xlab = "outcome")
         }
       , diabetes = diabetes)

#NB : une analyse multivariée pourrait être conduite via le package missMDA (prochain tutoriel)
```

2. Explorer le dispositif des données manquantes. On pourra utiliser la fonction `aggr` du package VIM pour l'analyse univariée, `CramerV` du package DescTools pour l'analyse bivariée, `MCA` du package FactoMineR pour l'analyse multidimensionnelle.

```
# dispositif des données manquantes
is.na(diabetes)

# analyse univariée
res.aggr <- aggr(diabetes)
str(res.aggr)
res.aggr$missings
cbind(res.aggr$tabcomb, res.aggr$percent)

# analyse bivariée
var.na <- which(res.aggr$missing$Count>0)
names(var.na) <- colnames(diabetes)[var.na]
pattern <- is.na(diabetes[, var.na])
matcram <- PairApply(pattern, CramerV)
PlotCorr(matcram)

# analyse multivariée
MCA(pattern)
```

3. Explorer la nature du mécanisme. On pourra utiliser les fonctions `marginmatrix` et `matrixplot` du package VIM pour les analyses bivariées.

```
# Nuage de points

# une variable complète (age) et une incomplète (skin.thick)
marginmatrix(diabetes[,c("Age","Skin.Thick")])
# rouge = individus avec valeur manquante sur Skin.Thick
# bleu = individus avec valeur observée sur Skin.Thick
# les individus incomplets sur la variable Skin.Thick semblent davantage agés

# deux variables incomplètes
marginmatrix(diabetes[,c("Skin.Thick","BMI")])
# rouge = individus avec valeur manquante sur l'autre variable
# bleu = individus avec valeur observée sur l'autre variable

# tous les couples
marginmatrix(diabetes[,-ncol(diabetes)],cex=.2,gap=0)

# matrixplot
matrixplot(diabetes,sortby = 8)
```

```r
# Analyse multivariée par ACM
 # discrétisation selon méthode des quantiles
d_bins <- discretize_get_bins(data=diabetes)
diabetes.cat <- discretize_df(diabetes, d_bins)
summary(diabetes.cat) # modalités rares
35/nrow(diabetes)

 # ACM
res.mca <- MCA(diabetes.cat, graph=FALSE, level.ventil = 0.04)# permet de ne pas ventiler BP
plot(res.mca, choix = "ind", invisible = "ind")
```

Imputation multiple

1. Utiliser la fonction `mice` du package mice pour imputer le jeu de données. Quels sont les paramètres (nombre de tableaux imputés, nombre d'itérations, modèles conditionnels) utilisés par défaut ?

```r
?mice
res.mice <- mice(diabetes, maxit = 50, printFlag = FALSE)
tableau1 <- complete(res.mice, 1)
summary(tableau1)
```

2. Vérifier la convergence de l'algorithme

```r
plot(res.mice)
```

3. A l'aide de la fonction `densityplot` du package mice, comparer les distributions des valeurs imputées et observées pour chacune des variables.

```r
densityplot(res.mice)
```

4. A l'aide de la fonction `marginmatrix`, comparer les distributions des valeurs imputées et observées pour le couple de variables `(Age, Skin.Thick)` sur le premier jeu de données imputé. On pourra utiliser la fonction `complete` pour obtenir ce tableau. Faire de même pour l'ensemble des couples de variables.

```r
diabetes.vim <- cbind.data.frame(complete(res.mice), is.na(diabetes))
head(diabetes.vim)
colnames(diabetes.vim) <- c(colnames(diabetes), paste0(colnames(diabetes),"_imp"))

# pour le couple Age-Skin.Thick
par(mfrow=c(1,2))
marginmatrix(diabetes[, c("Age", "Skin.Thick")])
marginmatrix(diabetes.vim[, c("Age", "Skin.Thick", "Age_imp", "Skin.Thick_imp")], delimiter = "_imp")

#Age en fonction de Skin.Thick : les points marrons sont des points imputés ;
# boites à moutaches verticales (pas de chgmt), horizontale (unique car tout est observé sur Age).
# Globalement, distribution similaire à la situation des cas-complets

marginmatrix(diabetes[, c("Insulin", "Glucose")])
marginmatrix(diabetes.vim[, c("Insulin", "Glucose", "Insulin_imp", "Glucose_imp")], delimiter = "_imp")
# Insulin en fonction de Glucose :
# jaune = imputé sur la variable Insuline, marron sur la variable Glucose

#pour tous les couples
marginmatrix(diabetes.vim, delimiter = "_imp")
```

5. Vérifier l'ajustement du modèle d'imputation en utilisant la fonction `overimpute` du package `micemd`. On pourra paralléliser les calculs en spécifiant l'argument `nnodes`.

```
#overimputation

nnodes <- detectCores()-1

res.mice.over <- mice.par(diabetes, m = 100, nnodes = nnodes, maxit = 20)

plotinds <- sample(seq(nrow(diabetes)), size = 30)
plotvar <- 6# correspond à la variable BMI
res.over <- overimpute(res.mice.over,
                       plotinds = plotinds,
                       plotvar = plotvar,
                       nnodes = nnodes)
```

5. Proposer d'autres modèles d'imputation pour la variable `BMI`.

```
res.mice$method
par(mfrow = c(2, 2), mar = c(4, 4, 2, 1) + 0.1)
for (method.imp in c("rf", "norm", "norm.boot")){
method <- res.mice$method
method["BMI"] <- method.imp
res.mice.tmp <- mice(diabetes, method = method, maxit = 20, printFlag = FALSE)
print(densityplot(res.mice.tmp,~ BMI,main=method.imp))

}

# On effectue l'overimputation

par(mfrow=c(1,2))

#norm
method <- res.mice$method
method["BMI"] <- "norm"
res.mice.over.norm <- mice.par(diabetes, m = 100, nnodes = nnodes, maxit = 20, method = method)

res.over.norm <- overimpute(res.mice.over.norm,
                       plotinds = plotinds,
                       plotvars = 6,
                       nnodes = nnodes)

#RF
method <- res.mice$method
method["BMI"] <- "rf"
res.mice.over.rf <- mice.par(diabetes, m = 100, nnodes = nnodes, maxit = 20, method = method)

res.over.rf <- overimpute(res.mice.over.rf,
                       plotinds = plotinds,
                       plotvars = 6,
                       nnodes = nnodes)
```

7. A l'aide de la fonction `with.mids`, ajuster un modèle de régression logistique sur chaque tableau imputé expliquant la variable `Outcome` à partir de l'ensemble des variables explicatives, puis agréger les résultats à l'aide de la fonction `pool`.

```r
class(res.mice)
fit <- with(res.mice, glm(Outcome~Preg+Glucose+BP+Skin.Thick+Insulin+BMI+DPF+Age,
                    family = binomial))

length(fit$analyses)
fit$analyses[[1]]

#pooling
res.pool <- pool(fit)
summary(res.pool)
```

8. Comparer avec une analyse des cas-complets.

```r
res.glm.cc <- glm(Outcome~., family=binomial,data=diabetes)
summary(res.glm.cc)
```

9. Effectuer une analyse de sensibilité par la méthode de l'ajustement delta sur la variable `Skin.Thick`. On pourra se reporter à la vignette dédiée (https://www.gerkovink.com/miceVignettes/Sensitivity_analysis/Sensitivity_analysis.html).

```r
boxplot(diabetes$Skin.Thick)
delta <- c(0, 10, 20, 50)
imp.all <- vector("list", length(delta))
names(imp.all)<-delta
post <- res.mice.over$post
for (i in 1:length(delta)){
  d <- delta[i]
  cmd <- paste("imp[[j]][,i] <- imp[[j]][,i] +", d)
  post["Skin.Thick"] <- cmd
  imp <- mice(diabetes, post = post, maxit = 5, seed = i, print = FALSE)
  imp.all[[i]] <- imp
}
par(mfrow=c(1,2))
bwplot(imp.all[["0"]])
bwplot(imp.all[["50"]])
densityplot(imp.all[["0"]], lwd = 3)
densityplot(imp.all[["50"]], lwd = 3)
```