

# Gestion des données manquantes par imputation multiple séquentielle

Vincent Audigier

11 Septembre 2024

## Importation des données

Téléchargez le dossier **Public** disponible [ici](#), puis décompressez le. Si vous travaillez sous Rstudio, ouvrez le fichier Rmd intitulé **TP\_enonce\_imputation\_multiple\_editable**. Dans le cas contraire, ouvrez R, puis spécifiez le dossier de travail approprié à l'aide de la fonction **setwd** selon le modèle suivant :

```
setwd("C:/Users/Audigier/Documents/Public/")# à modifier en fonction de l'endroit où sont décompressées
```

Chargez les données à l'aide de la fonction **load**

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

Les données portent sur 768 femmes adultes (âgées d'au moins 21 ans) de la tribu indienne Pima (Akimel O'odham) vivant près de Phoenix, Arizona, USA. Celles-ci sont décrites par 9 variables :

- **Preg** : nombre de grossesses
- **Glucose** : glycémie après un test de tolérance au glucose (mg/dL)
- **BP** : tension artérielle diastolique (mmHg)
- **Skin.Thick** : indice d'obésité (mm)
- **Insulin** : concentration en insuline (mIU/L)
- **BMI** : indice de masse corporelle (poids en kg/(taille en m)<sup>2</sup>)
- **DPF** : indice d'antécédents familiaux pour le diabète
- **Age** : âge en années
- **Outcome** : variable indicatrice de l'absence (0) ou présence (1) de diabète selon les critères de l'organisation mondiale de la santé

Une description plus complète du jeu de données est disponible [ici](https://github.com/niharikagulati/diabetesprediction) : <https://github.com/niharikagulati/diabetesprediction>

## Packages R

Installation des librairies requises (inutile si déjà installées)

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

Chargement des librairies (indispensable)

```
library(mice)
library(micemd)
library(FactoMineR)
library(parallel)
library(DescTools)
```

```
library(VIM)
library(funModeling)
```

## 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.

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

# variable cible et explicatives

## pour une variable
boxplot(diabetes[, "Age"] ~ diabetes$Outcome)

## pour toutes

par(mfrow = c(3, 3))
sapply(
  c("Preg", "Glucose", "BP", "Skin.Thick", "Insulin", "BMI", "DPF", "Age"),
  FUN = function(xx, diabetes) {
    boxplot(
      diabetes[, xx] ~ diabetes$Outcome,
      main = xx,
      ylab = xx,
      xlab = "outcome"
    )
  }, diabetes = diabetes
)

#NB : une analyse multivariée pourrait aussi être envisagée via le package missMDA
```

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

# 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)
par(mfrow = c(1, 1))
PlotCorr(matcram)
```

```
# analyse multivariée
res.mca <- MCA(pattern)
```

```
# zoom sur le graphique des modalités
plot.MCA(res.mca,
  choix = "ind",
  invisible = "ind",
  cex = .6)
```

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 âgés
```

```
# 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 = "Age")
```

```
# 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.MCA(
  res.mca,
  choix = "ind",
  invisible = "ind",
  cex = .7,
  selectMod = "cos2 25"
)
```

```
# association DPF < 0.220 et NA sur Insuline ?
```

```
marginmatrix(diabetes[, c("DPF", "Insulin")])
fisher.test(is.na(diabetes$Insulin), diabetes$DPF < 0.22)
```

## 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 ?
2. Vérifier la convergence de l'algorithme
3. A l'aide de la fonction `densityplot` du package `mice`, comparer les distributions des valeurs imputées et observées pour chacune des variables.
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.
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`.
6. Proposer d'autres modèles d'imputation pour la variable `Insulin`.
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`.
8. Comparer avec une analyse des cas-complets.