

# Gestion des données manquantes sous R

## Table des matières

<b>1. LES IDENTIFIER .....</b>	<b>2</b>
1.1. GENERALITES.....	2
1.2. DIFFERENTES SITUATIONS.....	2
1.2.1. <i>Les manquantes sont complètement au hasard</i> .....	2
1.2.2. <i>Les manquantes sont peut-être au hasard</i> .....	3
1.2.3. <i>Les manquantes ne sont pas au hasard</i> .....	3
<b>2. LES TRAITER.....</b>	<b>3</b>
2.1. LES CONSERVER OU LES ENLEVER .....	3
2.2. LES REMPLACER .....	5
2.2.1. <i>Utiliser un indicateur de données qualitatives manquantes</i> .....	5
2.2.2. <i>L'imputation simple</i> .....	5
2.2.3. <i>L'imputation multiple</i> .....	6
2.2.3.1. Cas général : package {mice} .....	6
2.2.3.2. Analyses multifactorielles : package {missMDA} .....	9
<b>3. RÉFÉRENCES .....</b>	<b>11</b>

## 1. LES IDENTIFIER

### 1.1. Généralités

Il convient d'être particulièrement attentif lors de l'échange de données entre différents logiciels, systèmes et usagers.

Les valeurs manquantes sont représentées dans R par la valeur NA - respecter les majuscules (Not Available). Cependant cette nomenclature n'est pas commune à tous les logiciels (par exemple une case vide dans Excel ou LibreOffice, ou bien un point dans SAS) et il peut arriver, lors d'une importation, que les données manquantes ne soient pas reconnues en tant que telles par R mais qu'elles soient assimilées à une modalité. Pour résoudre cela, il faut indiquer au logiciel R dès l'importation des données qu'elle est la valeur qui identifie les données manquantes. Ainsi dans le tableau de données importé toutes les données manquantes auront la valeur 'NA'.

Une fois les données manquantes correctement libellées, il existe différentes commandes permettant de les identifier :

**is.na** # détecte s'il y a des données manquantes.

**colSums(is.na())** # compte les données manquantes par colonne de l'objet désigné dans ().

**rowSums(is.na())** # compte les données manquantes par ligne de l'objet désigné dans ().

**na.fail()** # fournit un message d'erreur si l'objet désigné contient au moins une valeur "NA".

A contrario :

**complete.cases()** # identifie les lignes complètes qui n'ont pas du tout de données manquantes dans un tableau.

### 1.2. Différentes situations

Des traitements différents doivent être mis en œuvre selon la situation.

#### 1.2.1. Les manquantes sont complètement au hasard

Les manquantes peuvent être complètement au hasard. La probabilité qu'une observation soit incomplète est une constante, c'est à dire que le fait de ne pas avoir de valeur pour une variable ne dépend d'aucune autre variable.

Exemple : Supposons que l'on dispose d'une vingtaine de variables X1 à X20 parmi lesquelles X1 = âge ; X2 = sexe ; X3 = glycémie.

La probabilité que l'âge soit NA ne dépend ni du sexe ni de la glycémie. Elle est la même pour tous les sujets. Dans ce cas on procède simplement à l'analyse des données complètes, c'est à dire après avoir enlevé les individus présentant des NA. Les résultats ne sont pas biaisés mais il y a une perte de puissance et de précision car l'effectif est moindre.

### 1.2.2. Les manquantes sont peut-être au hasard

Les manquantes *peuvent être* au hasard. La probabilité qu'une valeur soit manquante ne dépend que de valeurs observées, c'est à dire que le fait de ne pas avoir de valeur pour certains individus pour une variable est dépendant d'une autre (ou d'autres) variables pour lesquelles on dispose des valeurs.

Exemple : on dispose de 20 variables dont X1 = âge ; X2 = sexe ; X3 = glycémie. La probabilité que la glycémie soit manquante ne dépend que de l'âge et du sexe, lesquels ne sont pas NA. Elle n'est pas la même pour tous les sujets. En supprimant les individus comportant des données manquantes on ne va conserver qu'une sous-population. Il convient donc de les remplacer (voir plus bas).

### 1.2.3. Les manquantes ne sont pas au hasard

Les manquantes ne sont pas au hasard. La probabilité qu'une observation soit incomplète dépend d'autres valeurs observées ou non, c'est à dire que le fait de ne pas avoir de valeurs pour une variable est lié au fait de ne pas en avoir non plus pour une ou plusieurs autres.

Exemple : on dispose de 20 variables dont X1 = âge ; X2 = sexe ; X3 = glycémie. La probabilité que l'âge soit NA dépend du fait que le sexe et/ou la glycémie sont également manquants. Les NA sont donc informatifs et il convient de les remplacer (voir plus bas).

## 2. LES TRAITER

Les données manquantes pourront être tout simplement ignorées mais aussi parfois remplacées.

### 2.1. Les conserver ou les enlever

Certaines fonctions autorisent l'argument **use = "complete.obs"**. Il permet par exemple lors du calcul d'une corrélation d'éliminer les observations (lignes) pour lesquelles l'une des deux valeurs (appartenant à la variable 1 ou à la variable 2) est manquante.

Exemple :

```
cor(fichier$var1, fichier$var2, use = "complete.obs")
```

Par défaut, R renvoie NA pour un grand nombre de calculs (sum, mean, median, var, ...) lorsque les données quantitatives comportent une valeur manquante car le calcul n'a pu se faire. On peut cependant modifier cette attribution en ajoutant l'argument 'na.rm' aux commandes de la fonction.

**na.rm = TRUE** (*non available removed*) indique à R de réaliser le calcul pour l'effectif total diminué des individus avec données manquantes.

**na.rm = FALSE**, au contraire, les conserve.

**Attention :** cet argument `na.rm` n'est pas présent dans toutes les fonctions de R. Certaines fonctions ne présentent pas de possibilité de gestion des données manquantes (par exemple la fonction `plsda()` du package `{mixOmics}`) et d'autres fonctions ont un autre argument pour les gérer. Par exemple la fonction `cor()` ci-dessous utilise un argument `'use ='` ou encore la fonction `mvr()` du package `{pls}` utilise l'argument `'na.action ='`.

`na.omit(x)` supprime les observations avec données manquantes de l'objet `x`, c'est-à-dire supprime les lignes correspondantes si `x` est une matrice ou un tableau de données.

`x <- x[!is.na(x)]` élimine les données manquantes de l'objet `x` qui doit-être un vecteur.

#### Un peu de pratique

- ✓ `data(iris)` # Charge les données 'iris'.
- ✓ Écrire la commande `fix(iris)` # Affiche les données dans l'éditeur.
- ✓ Dans l'éditeur effacer la valeur 5.1 (première ligne, première colonne, double clic) puis faire Entrée. La case n'est plus vide mais contient la valeur 'NA'.
- ✓ Fermer l'éditeur.
- ✓ Écrire les commandes suivantes et observer les résultats :

`is.na(Erreur ! Signet non défini.iris)` # Détection des données manquantes

```

      Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1             TRUE          FALSE          FALSE          FALSE  FALSE
2             FALSE          FALSE          FALSE          FALSE  FALSE
3             FALSE          FALSE          FALSE          FALSE  FALSE
etc.
```

`colSums(is.na(iris))` # Localisation des manquantes dans les colonnes

```

      Sepal.Length Sepal.Width Petal.Length Petal.Width      Species
               1              0              0              0              0
```

`na.fail(Erreur ! Signet non défini.iris$Sepal.Length)` # Renvoie un message d'erreur si la variable contient des données manquantes :

```
[1] ERREUR: missing values in object
```

`na.fail(iris$Sepal.Width)` # Renvoie toutes les valeurs de la variable si elle ne contient pas de manquantes.

`complete.cases(iris)` # Renvoie un vecteur logique indiquant TRUE s'il n'y a pas de valeurs manquantes.

```

[1] FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[13] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[25] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
etc.
```

`mean(iris$Sepal.Length)` # Renvoie la moyenne s'il n'y a pas de manquantes.

```
[1] NA
```

```
mean(iris$Sepal.Length, na.rm=TRUE) # On précise de ne pas utiliser les
individus ayant des manquantes.
```

```
[1] 5.848322
```

```
iris_nal <- na.omit(iris) # Construit une nouvelle base sans NA.
colSums(is.na(iris_nal)) # Demande la somme des manquantes dans
chaque colonne de la nouvelle base.
```

```
Sepal.Length  Sepal.Width Petal.Length  Petal.Width      Species
           0           0           0           0           0
```

```
cor(iris$Sepal.Length, iris$Sepal.Width) # Demande la
corrélation entre 2 variables alors que l'une d'elles comporte des NA.
```

```
[1] NA
```

```
cor(iris$Sepal.Length, iris$Sepal.Width,
    use = "complete.obs") # Demande la corrélation en enlevant les
individus qui ont des manquantes dans l'une ou l'autre variable.
```

```
[1] -0.1121059
```

## 2.2. Les remplacer

Plusieurs familles de méthodes sont possibles.

### 2.2.1. Utiliser un indicateur de données qualitatives manquantes

On remplace alors 'NA' par une nouvelle modalité, 'manq' par exemple. L'avantage est que l'analyse portera sur tous les individus. Cela suppose que les manquantes soient réparties au hasard ou complètement au hasard. Cette méthode permettra d'apprécier le risque de biais : une interaction significative entre l'indicateur des données manquantes et une variable explicative signalera un biais. Cette méthode ne protège donc pas contre le risque de biais.

### 2.2.2. L'imputation simple

L'imputation simple consiste à remplacer chaque NA par des données prédites ou simulées. Différentes méthodes d'imputation simple sont possibles :

- Dans le cas de mesures répétées, la prolongation de la dernière observation faite. On suppose que la valeur reste inchangée depuis la dernière mesure.
- La méthode du "plus proche voisin". La valeur manquante est remplacée par une valeur observée chez un individu ayant les mêmes caractéristiques. Plusieurs algorithmes existent pour trouver le "plus proche voisin", utilisant différentes fonctions de distance.
- Le remplacement par la moyenne (ou la médiane selon la taille des échantillons) des mesures disponibles. Toutes les NA d'une même variable auront la même valeur. Les estimations ne seront pas biaisées seulement si les données manquantes sont complètement au hasard.
- L'utilisation d'un modèle de régression peut permettre de remplacer une valeur manquante  $Y_i$  par une valeur prédite obtenue par régression de  $Y$  sur une ou plusieurs variables  $X$ . Cela a pour effet de sous-estimer la variance de  $Y$ .

### 2.2.3. L'imputation multiple

L'imputation multiple consiste à créer plusieurs valeurs possibles pour la valeur manquante. Cela permet de refléter correctement l'incertitude des NA, de préserver les aspects importants des distributions et de préserver les relations importantes entre les variables, bien que les valeurs manquantes ne soient pas forcément prédites avec grande précision.

#### 2.2.3.1. Cas général : package `{mice}`

La méthode consiste à remplacer la valeur manquante par  $m$  valeurs tirées au hasard d'une distribution appropriée. Cela permet de construire  $m$  bases de données complètes comportant les valeurs observées et les valeurs imputées. En combinant les résultats des  $m$  analyses faites sur ces bases de données complètes on évalue la variabilité supplémentaire due aux données manquantes. Plusieurs algorithmes permettent cette approche. Le plus utilisé repose sur la méthode MCMC (Monte Carlo Markov Chain) créant une chaîne de Markov convergeant vers la distribution prédictive *a posteriori* des données manquantes. Les méthodes d'imputation multiple sont les plus satisfaisantes.

#### Pratique de l'imputation multiple avec le package `{mice}` :

Le package `{mice}` ayant été installé, utiliser le menu 'Packages' / 'Charger le package' et choisir 'mice' (ou taper `library(mice)` et exécuter). Ce package contient un fichier de données 'nhanes' comportant 25 sujets et 4 variables :

- 'age' en 3 groupes d'âge, 1=20-39, 2=40-59, 3=60et+ ;
- 'bmi', index de masse corporelle (quantitative) ;
- 'hyp', hypertension, 1=non, 2=oui ;
- chl, cholestérol total (quantitative).

```
data(nhanes)
```

```
head(nhanes) # listage des 6 premières lignes
```

```
  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
5   1 20.4   1 113
6   3   NA  NA 184
```

```
# Proportion de données NA par variable
```

```
colSums(is.na(nhanes)) / nrow(nhanes)
```

```
  age  bmi  hyp  chl
0.00 0.36 0.32 0.40
```

```
# Proportion de données NA par individu
```

```
rowSums(is.na(nhanes)) / ncol(nhanes)
```

```
  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16
0.75 0.00 0.25 0.75 0.00 0.50 0.00 0.00 0.00 0.75 0.75 0.75 0.00 0.00 0.25 0.75
17  18  19  20  21  22  23  24  25
0.00 0.00 0.00 0.25 0.75 0.00 0.00 0.25 0.00
```

```
# Pattern des données manquantes
md.pattern(nhanes)
  age hyp bmi chl
13  1  1  1  1  0  <= 13 lignes complètes
  1  1  1  0  1  1
  3  1  1  1  0  1  <= 3 lignes où seule chl est NA
  1  1  0  0  1  2
  7  1  0  0  0  3
    0  8  9 10 27  <= total de NA = 27
```

# La fonction `mice()` : "multivariate imputation by chained equations".  
Arguments principaux \*:

- data : data frame
- m : nombre d'imputations multiples (m=5\*)
- imputation method : méthode utilisée
  - norm : régression linéaire bayésienne (numeric)
  - pmm\* : moyenne prédite par appariement (numeric)
  - mean : moyenne marginale (numeric)
  - logreg\* : régression logistique (2 catégories)
  - polyreg\* : régression logistique multinomiale ( $\geq 2$  cat.)
  - lda : analyse discriminante linéaire ( $\geq 2$  catégories)
  - seed : graine d'échantillonnage aléatoire à partir des données observées. (NA\*)

\* Valeurs par défaut.

N.B. D'autres méthodes d'imputation sont possibles. Voir l'aide de la fonction `mice()`.

```
imp <- mice(nhanes)
imp
Multiply imputed data set
Call:
mice(data = nhanes)
Number of multiple imputations: 5 <= valeur par défaut
Missing cells per column:
age bmi hyp chl
  0  9  8 10 <= récapitulatif des NA par colonne
Imputation methods:
  age  bmi  hyp  chl
    "" "pmm" "pmm" "pmm" <= méthodes par défaut
VisitSequence:
bmi hyp chl
  2  3  4 <= ordre d'utilisation des variables
PredictorMatrix:
  age bmi hyp chl
age  0  0  0  0
bmi  1  0  1  1 <=
hyp  1  1  0  1
chl  1  1  1  0
Random generator seed value: NA
```

indicateur des variables utilisées  
pour chaque variable avec des  
NA

# Données imputées pour la variable 'bmi'

**imp\$imp\$bmi**

	1	2	3	4	5
1	27.4	27.2	28.7	35.3	30.1
3	35.3	30.1	28.7	30.1	27.2
4	20.4	25.5	22.5	27.5	22.5
6	25.5	24.9	22.5	25.5	24.9
10	21.7	30.1	22.5	29.6	25.5
11	35.3	22.0	26.3	35.3	29.6
12	30.1	22.5	22.5	27.4	25.5
16	30.1	30.1	35.3	30.1	30.1
21	35.3	22.0	33.2	27.5	22.7

Lignes pour lesquelles bmi est NA.

1ère imputation    2ème imputation    5ème imputation

# Visualisation des 5 premières lignes du premier jeu de données complétées :

**complete(imp, 1)[1:5,]**

	age	bmi	hyp	chl
1	1	27.4	1	199
2	2	22.7	1	187
3	1	35.3	1	187
4	3	20.4	2	284
5	1	20.4	1	113

# Visualisation des 5 premières lignes du 5ème jeu de données complétées :

**complete(imp, 5)[1:5,]**

	age	bmi	hyp	chl
1	1	30.1	1	229
2	2	22.7	1	187
3	1	27.2	1	187
4	3	22.5	1	229
5	1	20.4	1	113

# Régression linéaire de chl sur age et hyp

**fit <- lm.mids(chl~age+hyp, imp)** <= commande de l'analyse.

**pool(fit)** <= pour obtenir les résultats des 5 analyses complètes.

Call: **pool(object = fit)**

Pooled coefficients: <= résultat final de la régression linéaire.

(Intercept)	age	hyp
148.05118	13.88849	18.09283

Fraction of information about the coefficients missing due to nonresponse:

(Intercept)	age	hyp	
0.2146295	0.2098358	0.1703749	<=fraction de l'information due aux non-réponses.



### 2.2.3.2. Analyses multifactorielles : package {missMDA}

Ce package est complémentaire à {FactoMineR}. Les méthodes d'imputation des données manquantes en AFC, ACM, AFM ou AFDM sont très clairement exposées par un didacticiel en vidéo de F. Husson et J. Josse (2013), en ligne à cette adresse : [www.youtube.com/watch?v=hQ6tDtg0tx0](http://www.youtube.com/watch?v=hQ6tDtg0tx0)

C'est de cet exposé que sont tirées les explications résumées ci-dessous.

#### Méthodes pour l'ACP

- a. ACP itérative. Dans une première étape une ACP est réalisée en n'utilisant que les données disponibles. Cela permet un premier calcul des axes factoriels. Les données manquantes sont ensuite remplacées par la moyenne de la variable à laquelle elles appartiennent et une seconde ACP est réalisée, produisant de nouveaux axes. Les projections des individus ayant des données manquantes (remplacées par les moyennes) sur ces axes permettent de préciser de nouvelles valeurs à imputer à la place des moyennes. Suite à l'imputation par ces nouvelles valeurs, une nouvelle ACP produira de nouveaux axes, permettant à leur tour grâce encore aux projections des individus à corriger, de disposer de nouvelles valeurs à imputer. Des ACP supplémentaires sont ainsi réalisées, précisant à chaque fois les valeurs à imputer. L'algorithme se poursuit jusqu'à convergence.

- b. ACP itérative régularisée (*Josse et al., 2009*). Suite à la stabilisation par la convergence obtenue, on remarque que la variabilité des données imputées est moins grande que celle des autres valeurs. Il en résulte un risque de surajustement, en particulier quand on estime beaucoup de variables par rapport au nombre d'individus ou quand il y a beaucoup de données manquantes. Le risque est alors de surestimer la force des relations entre les variables. Ce problème est également fréquent quand on dispose de données très bruitées.

Il est résolu par une méthode dite "régularisée", dont le principal objectif est de diminuer le nombre de valeurs propres à retenir pour l'ACP.

Cette opération de calcul d'un nombre de dimensions à conserver (*Josse et al., 2011*) est opérée en remplaçant tour à tour chaque valeur observée par une donnée manquante ; on procède alors pour chaque cas à des ACP itératives avec un nombre de dimensions variable jusqu'à ce que la valeur 'NA' soit très proche de la valeur observée. Comme ces itérations sont réalisées pour chacune des valeurs disponibles, cette opération est très coûteuse en temps de calcul.

Mise en œuvre avec les données 'orange' disponibles dans le package `{missMDA}`.

```
library(missMDA) # Chargement du package
data(orange) # Chargement des données
nb <- estim_ncpPCA(orange, ncp.max=5) # Estimation du nombre de
dimensions.
comp <- imputePCA(orange, ncp=nb, scale=TRUE) # Complète le
tableau.
res.pca <- PCA(comp$completeObs) # Réalise l'ACP sur le tableau
complété.
```

### Méthodes pour l'ACM.

Dans le cas des données qualitatives on utilise une extension de l'ACP régularisée en l'appliquant au tableau disjonctif. Quand une donnée qualitative est manquante dans le tableau d'origine, elle se retrouve manquante pour toutes les modalités de la variable dans le tableau disjonctif. Une première imputation des manquantes dans le tableau disjonctif consiste à les remplacer par la moyenne de la colonne, ce qui correspond à la proportion des individus ayant la modalité. Cette valeur est ensuite progressivement ajustée par une ACP itérative régularisée du tableau disjonctif avec des poids spécifiques pour les lignes et les colonnes utilisant les marges du tableau.

	V1	V2	V3	...
ind. 1	a	NA	g	...
ind. 2	NA	f	g	...
ind. 3	a	e	h	...
ind. 4	a	e	h	...
ind. 5	b	f	h	...
ind. 6	c	f	h	...
ind. 7	c	f	NA	...
...	...	...	...	...

	V1a	V1b	V1c	V2e	V2f	V3g	V3h	...
ind. 1	1	0	0	<b>0.71</b>	<b>0.29</b>	1	0	...
ind. 2	<b>0.12</b>	<b>0.29</b>	<b>0.59</b>	0	1	1	0	...
ind. 3	1	0	0	1	0	0	1	...
ind. 4	1	0	0	1	0	0	1	...
ind. 5	0	1	0	0	1	0	1	...
ind. 6	0	0	1	0	1	0	1	...
ind. 7	0	0	1	0	1	<b>0.37</b>	<b>0.63</b>	...
...	...	...	...	...	...	...	...	...

Si l'on devait remplacer la donnée manquante pour la variable 1 de l'individu 2 on choisirait la modalité 'c' car c'est la plus représentée ; pour la variable 2 de l'individu 1 on choisirait la modalité 'e' ; etc.

Remarque importante : dans le cas de l'ACM on n'impute pas les valeurs du tableau initial mais seulement celles du tableau disjonctif. Il en résulte qu'il sera nécessaire de faire l'ACM directement sur le tableau disjonctif. Cette possibilité est disponible dans le package `{FactoMineR}`. Si l'ACM était réalisée en conservant les données manquantes, les résultats seraient très fortement structurés par les individus ayant des données manquantes.

### Mise en œuvre de l'ACM régularisée avec les données 'vnf' :

```
library(missMDA) # Chargement du package.
data(vnf) # Chargement des données.
ncp <- estim_ncpMCA(vnf) # Estimation du nombre de composantes.
tab.disj <- imputeMCA(vnf, ncp=4) # Imputation dans le tableau
disjonctif en utilisant le nombre de composantes calculé.
```

```
res.mca <- MCA(vnf, tab.disj=tab.disj) # Réalisation de l'ACM sur le
tableau disjonctif complété.
```

### Méthodes pour l'AFM

Pour l'analyse factorielle multiple on utilisera la même méthode que pour l'ACP pour les données quantitatives et la même méthode que pour l'ACM pour les données qualitatives, en prenant en compte la structure en groupes des données.

Cet aspect est important car en AFM on rencontre parfois des dispositions très particulières des données manquantes : elles peuvent être nombreuses dans un groupe de variables mais pas dans un autre.

## 3. RÉFÉRENCES

P.D. Allison, (2001) Missing Data. *Thousand Oaks, CA:Sage*.

R. Giorgi. Traitement des données manquantes. *SESSTIM, Faculté de Médecine, Aix-Marseille Université, Marseille, France*. [https://sesstim.univ-amu.fr/sites/default/files/ressources\\_pedagogiques/traitementna-rg.pdf](https://sesstim.univ-amu.fr/sites/default/files/ressources_pedagogiques/traitementna-rg.pdf)

J. Josse, J. Pagès, and F. Husson. Gestion des données manquantes en analyse en composantes principales. *Journal de la Société Française de Statistique*, 150(2):28–51, 2009.

J. Josse and F. Husson. Multiple imputation in pca. *Advances in data analysis and classification*, 5(3):231–246, 2011

Logiciel utilisé : R 4.0.4.

Remarques et suggestions : <https://www.anastats.fr/equipe-et-contact/>