Recodage multiple

Arno Muller

2023-06-28

Résumé

Lors du nettoyage des données, on peut être amené à appliquer le même recodage à plusieurs variables aux modalités similaire (valeurs manquantes, Ne sait pas, Batterie de variables semblables, … ). Pour éviter de copier-coller un même code, on peut facilement automatiser le processus sur R.

**Fonctions utilisées dans la fiche**

| Packages | Fonctions |
| --- | --- |
| **dplyr** | mutate\_all - mutate\_if - mutate\_at - ifelse - select |
| **Base R** | as.data.frame - for - grep - assign - get - paste |

Il existe plusieurs solutions pour automatiser un recodage dans R, nous allons aborder certaines d’entre-elles mais cette liste est loin d’être exhaustive.

# Données d’exemple

On commence par créer un jeu de données, composé de 10 lignes et 10 colonnes, et qui contient des valeurs manquantes:

set.seed(1)  
df <- as.data.frame(matrix(sample(c(NA, 1:10), 100, replace = TRUE), 10))  
df

V1 V2 V3 V4 V5 V6 V7 V8 V9 V10  
1 8 NA 8 5 8 5 5 8 6 1  
2 3 4 8 9 7 7 6 6 6 9  
3 6 4 4 9 8 1 5 3 2 NA  
4 NA 9 4 5 6 1 7 6 5 10  
5 1 5 1 3 7 5 6 5 1 10  
6 6 9 9 3 5 5 10 NA 9 9  
7 10 6 8 9 9 NA NA 4 10 9  
8 1 8 NA 8 6 2 3 5 9 7  
9 10 4 3 6 2 2 7 NA 6 9  
10 2 4 2 5 9 7 8 8 2 4

# Avec R de base :

Dans le R de base, il est possible de recoder l’entiereté des occurences d’une certaines valeurs (ou les valeurs manquantes) pour l’ensemble du jeu de données.

## Sur l’ensemble des données

### Exemple 1 : Changer l’ensemble des valeurs manquantes

# Je fais un copie du jeu de donnée  
df2 <- df  
# Je remplace tous les NA par la valeurs 0  
df2[is.na(df2)] <- 0  
df2

V1 V2 V3 V4 V5 V6 V7 V8 V9 V10  
1 8 0 8 5 8 5 5 8 6 1  
2 3 4 8 9 7 7 6 6 6 9  
3 6 4 4 9 8 1 5 3 2 0  
4 0 9 4 5 6 1 7 6 5 10  
5 1 5 1 3 7 5 6 5 1 10  
6 6 9 9 3 5 5 10 0 9 9  
7 10 6 8 9 9 0 0 4 10 9  
8 1 8 0 8 6 2 3 5 9 7  
9 10 4 3 6 2 2 7 0 6 9  
10 2 4 2 5 9 7 8 8 2 4

### Exemple 2 : Remplacer une valeur par une autre

# Je fais une copie de l'original  
df2 <- df  
# Je remplace tous les 10 par la valeur "DIX"  
df2[df2==10] <- "DIX"  
df2

V1 V2 V3 V4 V5 V6 V7 V8 V9 V10  
1 8 NA 8 5 8 5 5 8 6 1  
2 3 4 8 9 7 7 6 6 6 9  
3 6 4 4 9 8 1 5 3 2 <NA>  
4 <NA> 9 4 5 6 1 7 6 5 DIX  
5 1 5 1 3 7 5 6 5 1 DIX  
6 6 9 9 3 5 5 DIX NA 9 9  
7 DIX 6 8 9 9 NA <NA> 4 DIX 9  
8 1 8 NA 8 6 2 3 5 9 7  
9 DIX 4 3 6 2 2 7 NA 6 9  
10 2 4 2 5 9 7 8 8 2 4

## Sur une sélection de variables

Les jeux de données peuvent être composés d’un grand nombre de variables et donc on peut-être amené à effectuer un recodage uniquement sur certaines variables.

L’exemple ci-dessous montre comment procéder à partir d’une sélection de variable stockée dans le vecteur *var*.

### Exemple 3 : Recoder une sélection de variables

# Je fais une copie de l'original  
df2 <- df  
# Je choisis certaines variables sur lesquelle on appliquera le recodage  
var <- c("V1","V7","V9")  
  
# Fonctionne aussi avec la position de la colonne  
# var <- c(2,5,6)  
  
# On crée une boucle qui s'activera uniquement pour les variables contenues   
# dans var  
  
for (i in var) {  
 df2[[i]] <- ifelse(df2[[i]] == 10, "DIX", df2[[i]])  
}  
  
  
df2

V1 V2 V3 V4 V5 V6 V7 V8 V9 V10  
1 8 NA 8 5 8 5 5 8 6 1  
2 3 4 8 9 7 7 6 6 6 9  
3 6 4 4 9 8 1 5 3 2 NA  
4 <NA> 9 4 5 6 1 7 6 5 10  
5 1 5 1 3 7 5 6 5 1 10  
6 6 9 9 3 5 5 DIX NA 9 9  
7 DIX 6 8 9 9 NA <NA> 4 DIX 9  
8 1 8 NA 8 6 2 3 5 9 7  
9 DIX 4 3 6 2 2 7 NA 6 9  
10 2 4 2 5 9 7 8 8 2 4

Seulement les variables V1, V7 et V9 ont été recodées. On remarque que des valeurs 10 sont présentes dans la variable V10.

# Avec le package dplyrdu tidyverse :

Les boucles peuvent être complexe à prendre en main, et pas forcément conseillées dans R. Le package dplyr propose un ensemble de fonctions permettant de recoder plusieurs variables avec seule une fonction.

## Sur l’ensemble des données

La fonction mutate\_all permet de recoder l’ensemble des variables du jeu de données.

### Exemple 4 : **mutate\_all**

On remplace l’ensemble de 7 par la valeur “SEPT”

library(dplyr)  
  
df2 <- df %>%   
 mutate\_all(~ifelse(. == 7, "SEPT", .))  
df2

V1 V2 V3 V4 V5 V6 V7 V8 V9 V10  
1 8 NA 8 5 8 5 5 8 6 1  
2 3 4 8 9 SEPT SEPT 6 6 6 9  
3 6 4 4 9 8 1 5 3 2 <NA>  
4 NA 9 4 5 6 1 SEPT 6 5 10  
5 1 5 1 3 SEPT 5 6 5 1 10  
6 6 9 9 3 5 5 10 NA 9 9  
7 10 6 8 9 9 <NA> <NA> 4 10 9  
8 1 8 NA 8 6 2 3 5 9 SEPT  
9 10 4 3 6 2 2 SEPT NA 6 9  
10 2 4 2 5 9 SEPT 8 8 2 4

* Le signe **~** est obligatoire, et permet d’indiquer qu’on applique la formule qui le suit à l’ensemble du jeu de données.
* Les **.** sont une spécificité de la syntaxe sous forme de **pipes** (%>%). Ils sont synonymes de la base de données en entrée (df) mais permet de tenir compte de toutes les transformations des pipes précédentes faites sur cette base.

## Sur une sélection de variables

Les fonctions mutate\_if et mutate\_at permettent d’appliquer un recodage mais uniquement sur des variables choisies par une ou plusieurs conditions.

### Exemple 5 : mutate\_if

Dans cet exemple, on va appliquer un recodage uniquement aux variables numériques. Pour l’exemple, on commence par créer une variable catégorielle *var\_categ*, sur laquelle le recodage ne s’appliquera pas.

df2 <- df %>%   
 mutate(var\_categ = rep(c("bonjour", "5"), 5)) %>%   
 select(var\_categ, everything())  
df2

var\_categ V1 V2 V3 V4 V5 V6 V7 V8 V9 V10  
1 bonjour 8 NA 8 5 8 5 5 8 6 1  
2 5 3 4 8 9 7 7 6 6 6 9  
3 bonjour 6 4 4 9 8 1 5 3 2 NA  
4 5 NA 9 4 5 6 1 7 6 5 10  
5 bonjour 1 5 1 3 7 5 6 5 1 10  
6 5 6 9 9 3 5 5 10 NA 9 9  
7 bonjour 10 6 8 9 9 NA NA 4 10 9  
8 5 1 8 NA 8 6 2 3 5 9 7  
9 bonjour 10 4 3 6 2 2 7 NA 6 9  
10 5 2 4 2 5 9 7 8 8 2 4

On utilise **mutate\_if**, pour changer la valeur 5 en CINQ, uniquement pour les variables numériques.

df2 <- df2 %>%   
 mutate\_if(is.numeric, ~ifelse(. == 5, "CINQ", .))  
df2

var\_categ V1 V2 V3 V4 V5 V6 V7 V8 V9 V10  
1 bonjour 8 <NA> 8 CINQ 8 CINQ CINQ 8 6 1  
2 5 3 4 8 9 7 7 6 6 6 9  
3 bonjour 6 4 4 9 8 1 CINQ 3 2 NA  
4 5 NA 9 4 CINQ 6 1 7 6 CINQ 10  
5 bonjour 1 CINQ 1 3 7 CINQ 6 CINQ 1 10  
6 5 6 9 9 3 CINQ CINQ 10 <NA> 9 9  
7 bonjour 10 6 8 9 9 <NA> <NA> 4 10 9  
8 5 1 8 NA 8 6 2 3 CINQ 9 7  
9 bonjour 10 4 3 6 2 2 7 <NA> 6 9  
10 5 2 4 2 CINQ 9 7 8 8 2 4

### Exemple 6 : **mutate\_at**

la fonction mutate\_at permet d’appliquer un recodage à un ensemble de variables définies par leur nom.

Ici, on applique le recodage uniquement aux variables qui continennent la chaîne de caractéres V1 dans leur nom de variables, on recode donc V1 et V10.

# Sélection des variables  
var <- grep("1", names(df), value=TRUE)  
var

[1] "V1" "V10"

# On recode les NA en VAL.MANQ pour les variables de var  
df2 <- df %>%   
 mutate\_at(var, ~ ifelse(is.na(.) == TRUE, "VAL.MANQ", .))  
df2

V1 V2 V3 V4 V5 V6 V7 V8 V9 V10  
1 8 NA 8 5 8 5 5 8 6 1  
2 3 4 8 9 7 7 6 6 6 9  
3 6 4 4 9 8 1 5 3 2 VAL.MANQ  
4 VAL.MANQ 9 4 5 6 1 7 6 5 10  
5 1 5 1 3 7 5 6 5 1 10  
6 6 9 9 3 5 5 10 NA 9 9  
7 10 6 8 9 9 NA NA 4 10 9  
8 1 8 NA 8 6 2 3 5 9 7  
9 10 4 3 6 2 2 7 NA 6 9  
10 2 4 2 5 9 7 8 8 2 4

# Automatisation de la création de nouvelles variables

Jusqu’ici nous avons vu comment recoder des variables existantes en changeant leurs valeurs, mais il est souvent préférable de créer de nouvelles variables pour éviter de perdre la variable originale.

### Exemple 7 : Avec une boucle **for** et **assign**

Dans cet exemple, nous allons choisir 3 variables à recoder (“V1”,“V3”,“V5”), mais au lieu de les écraser avec leurs nouvelles valeurs nous allons créer des copies de ces variables qui porteront le suffixe *bis*.

Si l’opération peut sembler facile aux premier abord, créer des nouveaux objets et leur affecter automatiquement une nouvelle valeur pose un problème majeur dans R : **Il n’est pas possible d’écrire une fonction devant un symbole d’affectation comme le = ou le <-**

Cela signifie que pour créer de nouvelles variables, on ne peut pas écrire :

# IMPOSSIBLE  
paste0(V1, "\_bis") <- ifelse(df$V1 == 8, "HUIT", df$V1)  
# IMPOSSIBLE  
df$paste0(V1, "\_bis")

Pour contourner ce problème, on utilise la fonction **assign()**, qui contient deux arguments : assign(NOM\_NV\_OBJ, valeurs\_prisent). On peut donc créer des variables avec des noms issues de fonctions comme paste. Dans notre cas on colle *ANCIEN\_NOM* et *bis*.

Pour chercher cet objet stocké dans l’environnement et en faire une variable, on utilise la fonction get() qui permet d’utiliser des fonctions comme paste() pour récupérer les valeurs des objets crées automatiquement.

# Copie de l'original  
df2 <- df  
# Sélecion des variables  
var <- c("V1","V3","V5")  
  
for (i in var) {  
   
 assign(paste0(i,"\_bis"), # Nom de l'objet  
 ifelse(df2[[i]] == 8, "HUIT", df2[[i]]) # Valeur qu'il prend  
 )  
   
 # On crée la variable à partir de l'objet crée au dessus  
 df2[paste0(i,"\_bis")] <- get(paste0(i,"\_bis"))  
  
}  
  
  
# On affiche la base avec les nouvelles variables en premières  
df2 %>%   
 select(contains("bis"), everything())

V1\_bis V3\_bis V5\_bis V1 V2 V3 V4 V5 V6 V7 V8 V9 V10  
1 HUIT HUIT HUIT 8 NA 8 5 8 5 5 8 6 1  
2 3 HUIT 7 3 4 8 9 7 7 6 6 6 9  
3 6 4 HUIT 6 4 4 9 8 1 5 3 2 NA  
4 <NA> 4 6 NA 9 4 5 6 1 7 6 5 10  
5 1 1 7 1 5 1 3 7 5 6 5 1 10  
6 6 9 5 6 9 9 3 5 5 10 NA 9 9  
7 10 HUIT 9 10 6 8 9 9 NA NA 4 10 9  
8 1 <NA> 6 1 8 NA 8 6 2 3 5 9 7  
9 10 3 2 10 4 3 6 2 2 7 NA 6 9  
10 2 2 9 2 4 2 5 9 7 8 8 2 4

On a bien crée automatiquement de nouvelles variables, tout en conservant les anciennes.