Recoder des variables avec ifelse() - if\_else - case\_when()

Arno Muller

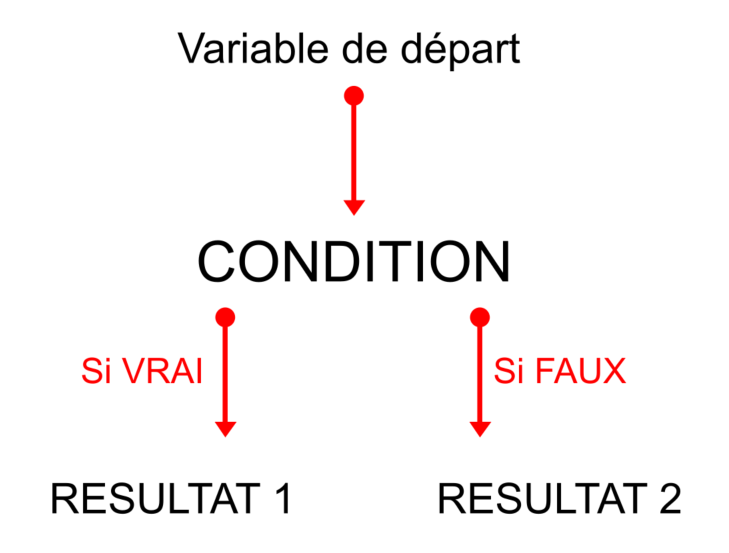
2023-06-27

Résumé

Lors d’un recodage d’une variable, on peut être emmené à créer des catégories dépendantes d’une ou plusieurs conditions. Pour cela, la plus part des logiciels permettent des fonctions qui suivent le schéma : SI condition ALORS catégorie1 SINON catégorie2. Sur R, plusieurs possibilités s’offrent à l’utilisateur.rice : notamment ifelse(), if\_else(), case\_when().

**Fonctions utilisées dans la fiche**

| Packages | Fonctions |
| --- | --- |
| **dplyr** | if\_else - case\_when - mutate |
| **Base R** | ifelse table |



# R base : ifelse()

## Pro/Cons :

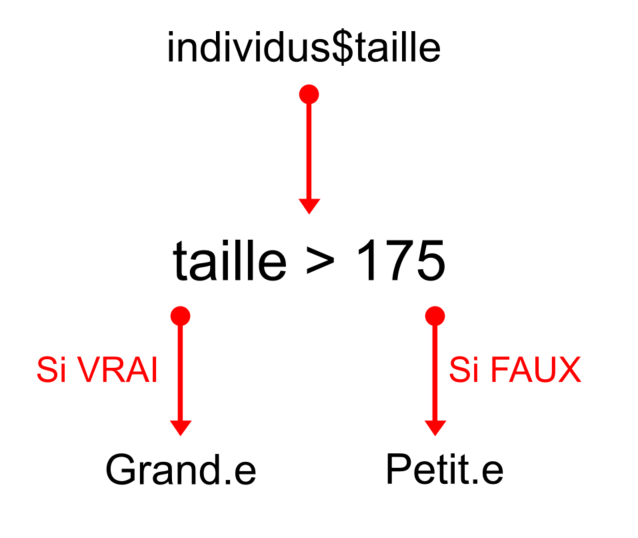
*Avantage :*  
- Pas besoin de packages supplémentaires

*Inconvénient :*

* Lent
* Problème de persistence des types de modalités (surtout avec les labels et les factors)[[1]](#footnote-23)
* Code moins lisible
* Gestion des conditions multiples illisibles

## Exemple 1 : Variables dichotomiques

Création d’une variable catégorielle *taille\_rec*, avec deux modalités : “Grand.e”, “Petit.e”.

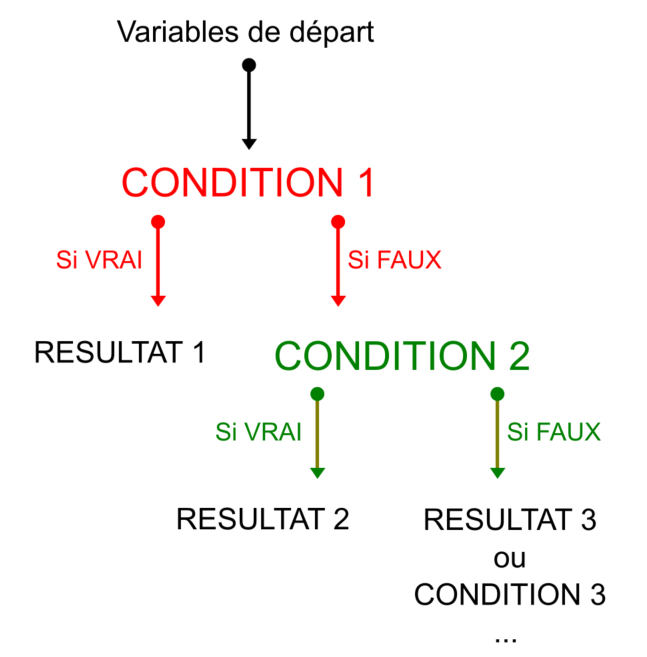


# Sélection d'une seed pour obtenir le même échantillon   
# dans la création des données fictive aléatoires  
set.seed(1)  
  
# Création du jeu de données exemple  
individus <- data.frame(  
 id = 1:20,  
 # Âges aléatoires entre 18 et 65 ans  
 age = sample(18:65, 20, replace = TRUE),   
 # Tailles aléatoires entre 150 et 200 cm  
 taille = sample(150:200, 20, replace = TRUE),   
 # Sexe aléatoire (Homme ou Femme)  
 sexe = sample(c("Homme", "Femme"), 20, replace = TRUE)   
)  
  
  
# Ajout d'une variable taill\_rec pour illustrer l'utilisation de ifelse()  
individus$taille\_rec <- ifelse(individus$taille > 175, # Condition   
 "Grand.e", # Catégorie si VRAI   
 "Petit.e") # Catégorie si FAUX  
  
  
# Affichage de la nouvelle variable  
table(individus$taille\_rec)

Grand.e Petit.e   
 12 8

## Exemple 2 : Variables avec plus de 2 modalités

Il est possible d’empiler les fonctions ifelse() pour créer des variables avec plus de deux modalités. Pour cela on suit le schéma suivant :



Dans cet exemple, on divise la taille en 3 catégories :

* Petit.e : < 165
* Moyen.ne : De 165 à 185
* Grand.e : > 185

individus$taille\_rec2 <- ifelse(individus$taille > 185, # Condition 1  
 "Grand.e", # Condi 1 Vrai   
 ifelse(individus$taille < 165,# Condition 2  
 "Petit.e", # Condi 2 Vrai  
 "Moyen.ne")) # Condi 1&2 Fausse  
   
  
  
# Affichage de la nouvelle variable  
table(individus$taille\_rec2)

Grand.e Moyen.ne Petit.e   
 8 9 3

## Exemple 3 : Conditions issues de deux variables initiales

Il est également possible de créer des variables à partir de conditions provenant de plusieurs variables.

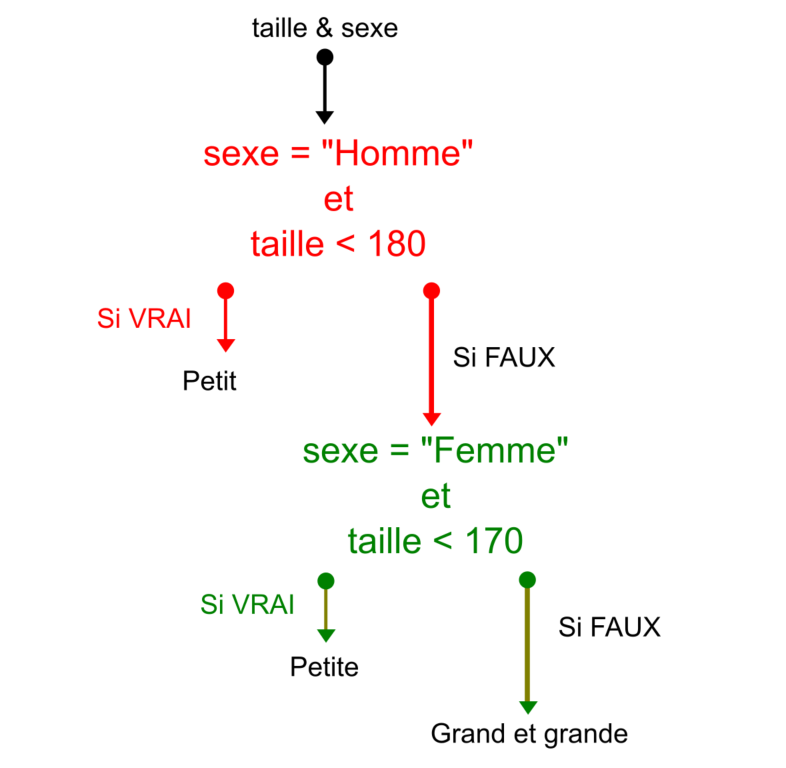
Dans cet exemple, nous allons créer une variable dont les catégories seront différentes selon pour les hommes et les femmes :

*Pour les hommes :*

* Petit.e : < 180
* Grand.e : >= 180

*Pour les femmes :*

* Petit.e : < 170
* Grand.e : >= 170



individus$taille\_rec3 <- ifelse(individus$sexe == "Homme" &   
 individus$taille < 180, # Condition 1  
 # Condition 1 Vraie :  
 "Petit.e",   
 # Sinon :  
 ifelse(individus$sexe == "Femme" &   
 individus$taille < 170, # Condition 2  
 # Condition 2 Vraie :  
 "Petit.e",   
 # Tous les autres :  
 "Grand.e"))   
   
  
  
# Affichage de la nouvelle variable  
table(individus$taille\_rec3)

Grand.e Petit.e   
 14 6

# Dplyr : if\_else()

## Pro/Cons :

*Avantage :*

Le package *dplyr*, présent dans le *tidyverse*, propose sa propre version de la fonction ifelse(), qui permet d’éviter certains problèmes rencontrer avec la version de base.

Ainsi, if\_else() s’utilise exactement de la même manière que ifelse, elle est plus rapide et permet de conserver le type de la variable initiale.

*Inconvénient :*

* Besoin d’installer un package
* Gestion des conditions multiples toujours illisibles

## Exemple 4 : if\_else()

#install.packages("dplyr")  
library(dplyr)  
  
individus$taille\_rec4 <- if\_else(individus$sexe == "Homme" &   
 individus$taille < 180, # Condition 1  
 # Condition 1 Vraie :  
 "Petit.e",   
 # Sinon :  
 if\_else(individus$sexe == "Femme" &   
 individus$taille < 170, # Condition 2  
 # Condition 2 Vraie :  
 "Petit.e",   
 # Tous les autres :  
 "Grand.e"))   
   
  
  
# Affichage de la nouvelle variable  
table(individus$taille\_rec4)

Grand.e Petit.e   
 14 6

**On obtient bien le même résultat**

# Dplyr : case\_when()

## Pro/Cons :

La gestion des conditions multiples pouvant être fastidieuse avec les fonctions ifelse(), le code devenant rapidemment illisible, *dplyr* propose la fonction case\_when() qui permet de l’organiser mieux.

Son fonctionnement se base sur des formules, il peut alors sembler plus ou moins compréhensibles selon les sensibilités de programmations.

# Exemple 5 : Variable dichotomique

library(dplyr)  
  
individus$taille\_rec5 <- case\_when(  
 individus$taille > 175 ~ "Grand", # Condition 1 ~ Resultat 1  
 individus$taille <= 175 ~ "Petit" # Condition 2 ~ Resultat 2  
)  
  
# Exemple avec catégorie "ELSE"  
individus$taille\_rec5bis <- case\_when(  
 individus$taille > 175 ~ "Grand", # Condition ~ Resultat   
 TRUE ~ "Petit" # Sinon ~ Resultat pour les autres cas  
)  
  
  
table(individus$taille\_rec5)

Grand Petit   
 12 8

table(individus$taille\_rec5bis)

Grand Petit   
 12 8

# Exemple 6 : Variable avec plusieurs modalités

On reprend l’exemple 3 avec case\_when().

library(dplyr)  
  
individus$taille\_rec6 <- case\_when(  
 individus$sexe == "Homme" & individus$taille < 180 ~ "Petit.e",  
 individus$sexe == "Femme" & individus$taille < 170 ~ "Petit.e",  
 TRUE ~ "Grand.e"  
)  
  
table(individus$taille\_rec6)

Grand.e Petit.e   
 14 6

# Attention !

## Ordre d’affectation des valeurs

Avec ifelse() ou avec case\_when, une fois la première condition remplie, la nouvelle valeur ne peux plus être modifiée, ainsi si une condition future se superpose, elle ne sera pas prise en compte dans le recodage. **On va du plus spécifique au plus général**

*Exemple :* On crée une variable taille avec comme modalité :

* Petit : < 170
* Moyen : 170 à 185
* Grand : > 185

individus <- individus %>%  
 select(id, taille) %>%   
   
 mutate(  
 # avec ifelse  
 var1 = ifelse(taille > 170, "Moyen",   
 ifelse(taille > 185, "Grand", "Petit")),  
   
 # avec if\_else  
 var2 = if\_else(taille > 170, "Moyen",   
 if\_else(taille > 185, "Grand", "Petit")),  
   
 # avec case\_when  
 var3 = case\_when(  
 taille > 170 ~ "Moyen",  
 taille > 185 ~ "Grand",   
 TRUE ~ "Petit"))   
  
table(individus$var1)

Moyen Petit   
 14 6

table(individus$var2)

Moyen Petit   
 14 6

table(individus$var3)

Moyen Petit   
 14 6

On voit qu’il n’y a pas la catégorie “Grand”, comme la valeur a déjà été affectée à la catégorie “Moyen”. Il faudrait alors :

individus <- individus %>%  
 mutate(var5 = case\_when(  
 taille > 185 ~ "Grand",   
 taille > 170 ~ "Moyen",  
 TRUE ~ "Petit"))   
  
table(individus$var5)

Grand Moyen Petit   
 8 6 6

## A propos des valeurs manquantes

Les fonctions ifelse() n’ont pas le même comportement pour le traitement des valeurs manquantes que la fonction case\_when.

Dans le cas des fonctions ifelse, si l’une des variables de conditions est un NA, le résultat sera également une valeur manquante dans la variable crée.

Dans le cas de la fonction case\_when, l’individu avec une valeur manquante aura comme nouvelle valeur la catégorie défini dans la partie “SINON”.[[2]](#footnote-47)

*Exemple :*

On crée un data frame avec des NA :

library(dplyr)  
df <- data.frame(a = c(1:3, NA, 4,NA,6:7), b = c(NA, letters[1:7]))  
df

a b  
1 1 <NA>  
2 2 a  
3 3 b  
4 NA c  
5 4 d  
6 NA e  
7 6 f  
8 7 g

On fait un recodage avec ifelse()

df %>%  
 mutate(res = if\_else(a > 3, "Yes",   
 if\_else(b == "c", "No",   
 if\_else(a > 5, "Maybe", "Done"))))

a b res  
1 1 <NA> <NA>  
2 2 a Done  
3 3 b Done  
4 NA c <NA>  
5 4 d Yes  
6 NA e <NA>  
7 6 f Yes  
8 7 g Yes

Les NA restent des NAs, alors qu’avec case\_when :

df %>%  
 mutate(res = case\_when(a > 3 ~ "Yes",   
 b == "c"~"No",   
 a > 5 ~ "Maybe",   
 TRUE ~ "Done"))

a b res  
1 1 <NA> Done  
2 2 a Done  
3 3 b Done  
4 NA c No  
5 4 d Yes  
6 NA e Done  
7 6 f Yes  
8 7 g Yes

On a pas de NA, ils ont été recodé avec la catégorie SINON.

**Il faut donc créer une condition avec is.na(variable)**

df %>%  
 mutate(res = case\_when(a > 3 ~ "Yes",   
 b == "c"~"No",   
 a > 5 ~ "Maybe",   
 is.na(a) | is.na(b) ~ NA,  
 TRUE ~ "Done"))

a b res  
1 1 <NA> <NA>  
2 2 a Done  
3 3 b Done  
4 NA c No  
5 4 d Yes  
6 NA e <NA>  
7 6 f Yes  
8 7 g Yes

1. <https://community.rstudio.com/t/case-when-why-not/2685/3> [↑](#footnote-ref-23)
2. <https://stackoverflow.com/questions/57807976/how-to-use-case-when-rather-then-if-else-error-in-my-code> [↑](#footnote-ref-47)