evaluation R

Samba

2025-02-25

## Charger les bibliothèques

1. Chargement des bases de données

dim(baseline\_mere)

## [1] 4256 17

#Partie 1 : Gestion et nettoyage des bases de données

1. Vérifions la présence de doublons dans les bases de données. Supprimons les doublons si nécessaire.

nrow(menage) - nrow(distinct(menage))

## [1] 1

nrow(baseline\_enf) - nrow(distinct(baseline\_enf))

## [1] 0

nrow(endline\_enf) - nrow(distinct(endline\_enf))

## [1] 0

nrow(baseline\_mere) - nrow(distinct(baseline\_mere))

## [1] 0

nrow(endline\_mere) - nrow(distinct(endline\_mere))

## [1] 0

# Afficher le nombre de doublons sur hhid  
menage %>%  
 group\_by(hhid) %>%  
 filter(n() > 1) %>%  
 summarise(nombre\_doublons = n()) %>%  
 arrange(desc(nombre\_doublons))

## # A tibble: 1 × 2  
## hhid nombre\_doublons  
## <chr> <int>  
## 1 4953484948 2

#Supprimons le doublon dans la bes menage  
menage\_uniq <- menage %>%  
 distinct(hhid, .keep\_all = TRUE)

# Fonction pour vérifier les doublons  
verif\_doublon <- function(data, key) {  
 total\_dups <- nrow(data) - nrow(distinct(data)) # Doublons complets  
 id\_dups <- data %>% group\_by(across(all\_of(key))) %>% filter(n() > 1) %>% nrow() # Doublons sur l'ID  
  
 return(list(  
 total\_dups = total\_dups,  
 id\_dups = id\_dups  
 ))  
}

# Vérification des doublons  
doublon\_baseline\_mere <- verif\_doublon(baseline\_mere, "hhid")  
doublon\_endline\_mere <- verif\_doublon(endline\_mere, "hhid")  
doublon\_baseline\_enf <- verif\_doublon(baseline\_enf, "hhid")  
doublon\_endline\_enf <- verif\_doublon(endline\_enf, "hhid")  
doublon\_menage <- verif\_doublon(menage, "hhid")

# Afficher les résultats  
list(  
 "Baseline mère" = doublon\_baseline\_mere,  
 "Endline mère" = doublon\_endline\_mere,  
 "Baseline enfant" = doublon\_baseline\_enf,  
 "Endline enfant" = doublon\_endline\_enf,  
 "Ménage" = doublon\_menage  
)

## $`Baseline mère`  
## $`Baseline mère`$total\_dups  
## [1] 0  
##   
## $`Baseline mère`$id\_dups  
## [1] 4256  
##   
##   
## $`Endline mère`  
## $`Endline mère`$total\_dups  
## [1] 0  
##   
## $`Endline mère`$id\_dups  
## [1] 4256  
##   
##   
## $`Baseline enfant`  
## $`Baseline enfant`$total\_dups  
## [1] 0  
##   
## $`Baseline enfant`$id\_dups  
## [1] 4256  
##   
##   
## $`Endline enfant`  
## $`Endline enfant`$total\_dups  
## [1] 0  
##   
## $`Endline enfant`$id\_dups  
## [1] 4256  
##   
##   
## $Ménage  
## $Ménage$total\_dups  
## [1] 1  
##   
## $Ménage$id\_dups  
## [1] 2

1. Assurez-vous que les noms des variables sont cohérents entre les bases de données Baseline et Endline (par exemple, les noms des nutriments doivent être identiques). Si ce n’est pas le cas, apportez des corrections.

# Afficher les noms des colonnes  
colnames(baseline\_mere)

## [1] "regionid" "communeid" "villageid" "hhid" "round"   
## [6] "s1\_q0" "s1\_q1" "s1\_q2" "V1" "protein\_g"   
## [11] "lipid\_tot\_g" "calcium\_mg" "iron\_mg" "V9" "vit\_b6\_mg"   
## [16] "vit\_b12\_mcg" "vit\_c\_mg"

colnames(endline\_mere)

## [1] "regionid" "communeid" "villageid" "hhid" "round"   
## [6] "s1\_q0" "s1\_q1" "s1\_q2" "energ\_kcal" "protein\_g"   
## [11] "lipid\_tot\_g" "calcium\_mg" "iron\_mg" "zinc\_mg" "vit\_b6\_mg"   
## [16] "vit\_b12\_mcg" "vit\_c\_mg"

# Renommer les variables si nécessaire  
endline\_mere <- endline\_mere %>%  
 rename\_with(~ colnames(baseline\_mere), everything())

colnames(baseline\_enf)

## [1] "regionid" "communeid" "villageid" "hhid" "round"   
## [6] "s1\_q0" "s1\_q1" "s1\_q2" "energ\_kcal" "protein\_g"   
## [11] "lipid\_tot\_g" "calcium\_mg" "iron\_mg" "zinc\_mg" "vit\_b6\_mg"   
## [16] "vit\_b12\_mcg" "vit\_c\_mg"

colnames(endline\_enf)

## [1] "regionid" "communeid" "villageid" "hhid" "round"   
## [6] "s1\_q0" "s1\_q1" "s1\_q2" "energ\_kcal" "protein\_g"   
## [11] "lipid\_tot\_g" "calcium\_mg" "iron\_mg" "zinc\_mg" "vit\_b6\_mg"   
## [16] "vit\_b12\_mcg" "vit\_c\_mg"

# Renommer les variables si nécessaire  
baseline\_enf <- baseline\_enf %>%  
 rename\_with(~ colnames(endline\_enf), everything())

1. Certaines variables peuvent contenir des données manquantes, cependant suivant une logique bien établie, il est possible de corriger ces données manquantes. Veuillez vérifier soigneusement les données et corriger les données manquantes de certaines variables si possibles.

colSums(is.na(menage))

## regionid communeid villageid hhid hhsize   
## 0 1 0 0 0   
## poly hh\_primary s1\_q2 s1\_q4a s2\_q1   
## 0 0 0 0 0   
## s2\_q2 s2\_q4 s29\_q1 demgrp1 demgrp2   
## 0 0 0 0 0   
## demgrp3 demgrp4 demgrp5 dependencyratio hfias\_score   
## 0 0 0 0 0   
## T1   
## 0

colSums(is.na(baseline\_enf))

## regionid communeid villageid hhid round s1\_q0   
## 0 0 0 0 0 0   
## s1\_q1 s1\_q2 energ\_kcal protein\_g lipid\_tot\_g calcium\_mg   
## 0 0 386 386 386 386   
## iron\_mg zinc\_mg vit\_b6\_mg vit\_b12\_mcg vit\_c\_mg   
## 386 386 386 386 386

colSums(is.na(baseline\_mere))

## regionid communeid villageid hhid round s1\_q0   
## 4 8 0 0 0 0   
## s1\_q1 s1\_q2 V1 protein\_g lipid\_tot\_g calcium\_mg   
## 0 0 673 673 673 673   
## iron\_mg V9 vit\_b6\_mg vit\_b12\_mcg vit\_c\_mg   
## 673 673 673 673 673

colSums(is.na(endline\_enf))

## regionid communeid villageid hhid round s1\_q0   
## 0 0 0 0 0 0   
## s1\_q1 s1\_q2 energ\_kcal protein\_g lipid\_tot\_g calcium\_mg   
## 0 0 532 532 532 532   
## iron\_mg zinc\_mg vit\_b6\_mg vit\_b12\_mcg vit\_c\_mg   
## 532 532 532 532 532

colSums(is.na(endline\_mere))

## regionid communeid villageid hhid round s1\_q0   
## 0 0 0 0 0 0   
## s1\_q1 s1\_q2 V1 protein\_g lipid\_tot\_g calcium\_mg   
## 0 0 664 664 664 664   
## iron\_mg V9 vit\_b6\_mg vit\_b12\_mcg vit\_c\_mg   
## 664 664 664 664 664

On peut remarquerv la presence de nombreuses données manquantes dans beaucoup de variables et ceci dans toutes les bases baseline et endline mais pour la base menage on remarque une seule valeur manquante dans la variable “communeid”.

Je vais choisi la methode de la médiane pour remplacer les valeurs manquantes (NA) car elle est plus robuste que la moyenne face aux valeurs aberrantes et fonctionne bien même lorsque les données sont asymétriques. Contrairement à la moyenne, qui peut être influencée par des valeurs extrêmes, la médiane représente mieux la tendance centrale des données, en particulier pour les variables nutritionnelles comme la consommation d’énergie et de nutriments. De plus, elle reste fiable même lorsque les valeurs sont très dispersées, contrairement au mode, qui peut être peu pertinent en présence de nombreuses valeurs uniques.

replace\_na\_par\_median <- function(data, cols) {  
 data <- data %>%  
 mutate(across(all\_of(cols), ~ ifelse(is.na(.), median(., na.rm = TRUE), .)))  
 return(data)  
}