# RÉPUBLIQUE DU SÉNÉGAL


**Un Peuple - Un But - Une Foi**

## Ministère de l'Économie, du Plan et de la Coopération

RÉPUBLIQUE DU SÉNÉGAL
*Un Peuple - Un But - Une Foi*
MINISTÈRE DE L'ÉCONOMIE
DU PLAN ET DE LA COOPÉRATION

**Agence Nationale de la Statistique et de la Démographie (ANSD)**


**École nationale de la Statistique et de l'Analyse économique Pierre Ndiaye (ENSAE)**


## PROJET STATISTIQUES SOUS R

# TP4 : Récupération des codes Admin 3 du Bénin et du Sénégal

---

**Rédigé par :**
**Mame Balla BOUSSO**
**Ameth FAYE**
**EDIMA Biyenda Hildegarde**
**Papa Amadou NIANG**
*Elèves ingénieurs statisticiens économistes*

**Sous la supervision de :**
**M. Aboubacar HEMA**
*ANALYSTE DE RECHERCHE CHEZ IFPRI*

**Année scolaire : 2024/2025**

# Introduction

Ce document présente le processus de fusion des bases de données afin de récupérer les codes admin 3 pour chaque commune enquêtée dans la base EHCVM.
Nous travaillons sur deux pays : le **Bénin** et le **Sénégal**.
L'approche utilisée pour le Bénin est basée sur un nettoyage simple et une fusion directe, tandis que pour le Sénégal, nous proposons une démarche interactive permettant d'associer les communes en comparant leur similarité textuelle.

# Librairies Utilisées

Nous utilisons les librairies suivantes :

- **dplyr** : pour la manipulation des données (filtrage, transformation, fusion, etc.).
- **stringi** et **stringr** : pour le traitement et la manipulation des chaînes de caractères (nettoyage, remplacement, etc.).
- **sf** : pour la manipulation des données spatiales (lecture des shapefiles, analyses géographiques).
- **haven** : pour la lecture des fichiers de données au format Stata (.dta).
- **labelled** : pour convertir les variables possédant des labels en facteurs.
- **readxl** : pour la lecture des fichiers Excel.

```
library(dplyr)
library(stringi)
library(stringr)
library(sf)
library(haven)
library(labelled)
library(readxl)
```

# I : Le Bénin

## Présentation du processus pour le Bénin

Pour le Bénin, nous disposons de trois sources de données : 1. **Shapefile** : Contient les limites administratives (communes). 2. **Fichier Excel** : Renferme des informations complémentaires, notamment les codes admin. 3. **Base EHCVM (Stata)** : Contient les données des enquêtes individuelles.

**Objectif** : Fusionner ces sources pour associer à chaque commune de la base EHCVM le code admin extrait des données géographiques.

### Étape 1 : Chargement et préparation des données

Nous commençons par charger l'ensemble des données.

```
# Chargement des données du Bénin

# Shapefile des communes
data_BEN <- st_read("data/Benin/shapefiles/geoBoundaries-BEN-ADM2.shp")
```

```
## Reading layer `geoBoundaries-BEN-ADM2' from data source
##   `D:\Projet statistique sous R\Cours\TP4_Communes\data\Benin\shapefiles\geoBound
BEN-ADM2.shp'
##   using driver `ESRI Shapefile'
## Simple feature collection with 77 features and 5 fields
## Geometry type: POLYGON
## Dimension:     XY
## Bounding box:  xmin: 0.774575 ymin: 6.225748 xmax: 3.851701 ymax: 12.40861
## Geodetic CRS:  WGS 84
```

```
# Fichier Excel avec les codes admin
data_BEN_Pcode <- read_excel("data/Benin/shapefiles/ben_adminboundaries_tabulardata
                             sheet = "ADM2")

# Base EHCVM (format Stata)
data <- read_dta("data/Benin/ehcvm/ehcvm_individu_ben2021.dta")

# Conversion des variables labellisées en facteurs pour faciliter le merge
data <- to_factor(data)
```

# Étape 2 : Nettoyage des noms de communes

Pour assurer une correspondance correcte entre les bases, nous normalisons les noms en : - Remplaçant les caractères accentués (É, È, é, è) par leur équivalent non accentué. - Convertissant tous les noms en minuscules.

```
# Remplacer les caractères accentués par leur équivalent sans accent
data$commune <- gsub("[ÉÈéè]", "e", data$commune)
data <- data %>% mutate(commune = tolower(commune))

# Pour le shapefile, on met en minuscule le nom des communes
data_BEN <- data_BEN %>% mutate(shapeName = tolower(shapeName))
data_BEN_Pcode <- data_BEN_Pcode %>% mutate(ADM2_FR = tolower(ADM2_FR))
```

# Étape 3 : Fusion des données géographiques

Nous fusionnons le shapefile avec le fichier Excel afin d'associer les informations complémentaires (codes admin) aux limites géographiques.

```
data_BEN_fin <- merge(data_BEN, data_BEN_Pcode, by.x = "shapeName", by.y = "ADM2_F
```

# Étape 4 : Fusion finale avec la base EHCVM

Enfin, nous associons les données géographiques (avec le code admin) aux enregistrements de la base EHCVM.

```
data_merge_BEN <- right_join(data, data_BEN_fin, by = c("commune" = "shapeName"))

# Affichage d'un aperçu du résultat
head(data_merge_BEN)
```

```
## # A tibble: 6 x 77
##   country  year vague  hhid grappe menage numind zae       zaemil departement
##   <chr>   <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl> <fct>      <fct>  <fct>
```

```
## 1 BEN     2021    2  1005     1     5     1 Transition,~ Trans~ alibori
## 2 BEN     2021    2  1005     1     5     2 Transition,~ Trans~ alibori
## 3 BEN     2021    2  1005     1     5     3 Transition,~ Trans~ alibori
## 4 BEN     2021    2  1005     1     5     4 Transition,~ Trans~ alibori
## 5 BEN     2021    2  1005     1     5     5 Transition,~ Trans~ alibori
## 6 BEN     2021    2  1005     1     5     8 Transition,~ Trans~ alibori
## # i 67 more variables: commune <chr>, arrondissement <fct>, milieu <fct>,
## #   hhweight <dbl>, resid <fct>, sexe <fct>, age <dbl>, lien <fct>,
## #   mstat <fct>, religion <fct>, ethnie <fct>, nation <fct>, agemar <dbl>,
## #   ea <chr>, em <chr>, ej <chr>, mal30j <fct>, aff30j <fct>, arrmal <dbl>,
## #   durarr <fct>, con30j <fct>, hos12m <fct>, couvmal <dbl>, moustiq <dbl>,
## #   handit <fct>, handig <fct>, serviceconsult <fct>, persconsult <fct>,
## #   alfa <dbl>, alfa2 <dbl>, scol <dbl>, educ_scol <fct>, educ_hi <fct>, ...
```

Si, dans la base EHCVM, une commune est identifiée sous le nom “cotonou”, la procédure ci-dessus permettra de lui associer le code admin présent dans le shapefile fusionné.

# II : Le Sénégal

## Présentation du processus pour le Sénégal

Pour le Sénégal, les noms de communes dans les différentes sources peuvent présenter des variations (abréviations, accents, fautes d’orthographe).
**Objectif** : Proposer une démarche interactive afin d’associer chaque commune du shapefile à celle de la base EHCVM en s’appuyant sur un score de similarité calculé sur la base de la distance de Levenshtein.

## Étape 1 : Définition des fonctions de nettoyage et de similarité

**1.1 Fonction de nettoyage des noms de communes** Cette fonction transforme les noms en minuscules, retire les préfixes inutiles (par exemple «COM.»), enlève les accents et supprime les espaces pour faciliter la comparaison.

```
nettoyage_commune <- function(x) {
  x %>%
    tolower() %>%
    str_replace_all("^(com\\.?\\s*|com\\s+|commune\\s*(de)?\\s*)", "") %>%
    stri_trans_general("Latin-ASCII") %>%
    str_replace_all("['’‘´`]", "") %>%
    str_replace_all("[^[:alnum:] ]+", " ") %>%
    str_squish() %>%
    str_replace_all(" ", "")
}

# Exemple d'utilisation
exemple_commune <- "COM. SénéGalKèmo"
cat("Avant nettoyage : ", exemple_commune, "\n")
```

```
## Avant nettoyage :  COM. SénéGalKèmo
```

```r
cat("Après nettoyage : ", nettoyage_commune(exemple_commune), "\n")
```

```
## Après nettoyage :  senegalkemo
```

**1.2 Fonction de calcul de la distance de Levenshtein** Cette fonction calcule la distance minimale (nombre d'opérations) nécessaire pour transformer une chaîne en une autre.

```r
levenshtein_distance <- function(s, t) {
  s_chars <- unlist(strsplit(s, split = ""))
  t_chars <- unlist(strsplit(t, split = ""))
  m <- length(s_chars)
  n <- length(t_chars)
  d <- matrix(0, nrow = m + 1, ncol = n + 1)
  for(i in 1:(m + 1)) { d[i, 1] <- i - 1 }
  for(j in 1:(n + 1)) { d[1, j] <- j - 1 }
  for(i in 1:m) {
    for(j in 1:n) {
      cost <- ifelse(s_chars[i] == t_chars[j], 0, 1)
      d[i + 1, j + 1] <- min(d[i, j + 1] + 1,
                             d[i + 1, j] + 1,
                             d[i, j] + cost)
    }
  }
  return(d[m + 1, n + 1])
}

# Exemple d'utilisation
s1 <- "chat"
s2 <- "chats"
cat("Distance de Levenshtein entre '", s1, "' et '", s2, "' : ",
    levenshtein_distance(s1, s2), "\n", sep = "")
```

```
## Distance de Levenshtein entre 'chat' et 'chats' : 1
```

**1.3 Fonction de calcul de la similarité normalisée** Cette fonction utilise la distance de Levenshtein pour calculer un score de similarité entre 0 et 1.

```r
similarity <- function(s, t) {
  d <- levenshtein_distance(s, t)
  max_len <- max(nchar(s), nchar(t))
  if(max_len == 0) return(1)
  return(1 - d / max_len)
}

# Exemple d'utilisation
cat("Similarité entre '", s1, "' et '", s2, "' : ",
    similarity(s1, s2), "\n", sep = "")
```

```
## Similarité entre 'chat' et 'chats' : 0.8
```

**1.4 Fonction pour calculer la matrice de similarité** Cette fonction construit une matrice dans laquelle chaque case correspond au score de similarité entre un nom issu du shapefile et un nom de la base EHCVM.

```r
compute_similarity_matrix <- function(shape_communes, ehcvm_communes, sim_func = s
  sim_matrix <- matrix(0, nrow = length(shape_communes), ncol = length(ehcvm_commun
```

```r
  rownames(sim_matrix) <- shape_communes
  colnames(sim_matrix) <- ehcvm_communes

  for (i in seq_along(shape_communes)) {
    for (j in seq_along(ehcvm_communes)) {
      sim_matrix[i, j] <- sim_func(shape_communes[i], ehcvm_communes[j])
    }
  }
  return(sim_matrix)
}

# Exemple d'utilisation
exemple_shape <- c("ruffisque", "pikine")
exemple_ehcvm <- c("ruffisque", "pikine", "dakar")
exemple_matrix <- compute_similarity_matrix(exemple_shape, exemple_ehcvm)
print(exemple_matrix)
```

```
##           ruffisque    pikine     dakar
## ruffisque 1.0000000 0.2222222 0.0000000
## pikine    0.2222222 1.0000000 0.1666667
```

# Étape 2 : Fusion interactive des bases

La fonction ci-dessous propose une méthode interactive qui calcule la similarité entre les noms de communes et vous permet de valider ou d'ajuster manuellement les correspondances lorsque le score de similarité est faible.

> **Note :** Cette fonction utilise `readline()` pour récupérer des réponses en temps réel. Elle est idéale pour une exécution interactive (par exemple dans RStudio) mais nécessitera des adaptations pour un rendu non-interactif.

```r
merge_bases_commune_interactive <- function(data_SEN, data, threshold = 0.8,
                                            top_n = 5) {

  # Extraction et nettoyage des noms de communes
  shape_communes <- data_SEN$ADM3_FR_clean
  ehcvm_communes <- unique(data$commune_clean)

  # Calcul de la matrice de similarité
  cat("Calcul des scores de similarité...\n")
  sim_matrix <- compute_similarity_matrix(shape_communes, ehcvm_communes)

  # Séparation en deux groupes :
  # Groupe A : score maximum >= threshold (correspondances automatiques)
  # Groupe B : score maximum < threshold (à valider manuellement)
  groupA_indices <- which(apply(sim_matrix, 1, max) >= threshold)
  groupB_indices <- setdiff(seq_along(shape_communes), groupA_indices)

  # --- Groupe A : correspondances automatiques ---
  cat("\n=== Groupe A : Correspondances automatiques (score >= ",
      threshold*100, "%) ===\n", sep = "")
  auto_mapping <- data.frame(
    shape_commune = shape_communes[groupA_indices],
    ehcvm_commune = sapply(groupA_indices, function(i) {
      candidates <- ehcvm_communes[order(sim_matrix[i, ], decreasing = TRUE)]
```

```
    return(candidates[1])
  }),
  score = sapply(groupA_indices, function(i) max(sim_matrix[i, ])),
  stringsAsFactors = FALSE
)

print(auto_mapping)

rep_auto <- readline(prompt = "\nValidez-vous toutes ces correspondances
                    automatiques ? (o/n) : ")
if(tolower(rep_auto) == "o") {
  mapping_A <- auto_mapping
} else {
  cat("\nEntrez les indices (dans la liste ci-dessus) des communes à modifier,
      séparés par une virgule\n(ou appuyez sur Entrée pour conserver celles non
      modifiées) : ")
  modif <- readline()
  mapping_A <- auto_mapping  # on part de la proposition automatique
  if(nchar(trimws(modif)) > 0) {
    modif_indices <- as.numeric(unlist(strsplit(modif, ",")))
    if(any(is.na(modif_indices)) || any(modif_indices < 1) ||
       any(modif_indices > nrow(auto_mapping))) {
      cat("Indices invalides. Aucune modification ne sera effectuée pour
          le Groupe A.\n")
    } else {
      for(mi in modif_indices) {
        current_shape <- auto_mapping$shape_commune[mi]
        i <- which(shape_communes == current_shape)[1]
        candidate_order <- order(sim_matrix[i, ], decreasing = TRUE)
        candidates <- ehcvm_communes[candidate_order]
        candidate_scores <- sim_matrix[i, candidate_order]
        n_candidates <- min(top_n, length(candidates))
        cat(sprintf("\nPour la commune '%s', voici les %d
                    meilleures correspondances :\n",
                    current_shape, n_candidates))
        for (k in 1:n_candidates) {
          cat(sprintf("  %d: '%s' (score = %.2f)\n", k,
                      candidates[k], candidate_scores[k]))
        }
        rep_choice <- as.numeric(readline(prompt = "Choisissez le
                                          numéro correspondant (ou 0 pour aucune
                                          correspondance) : "))
        if(!is.na(rep_choice) && rep_choice >= 1 && rep_choice <= n_candidates) {
          mapping_A$ehcvm_commune[mi] <- candidates[rep_choice]
          mapping_A$score[mi] <- candidate_scores[rep_choice]
        } else {
          mapping_A$ehcvm_commune[mi] <- NA
          mapping_A$score[mi] <- NA
          cat("Aucune correspondance validée pour cette commune.\n")
        }
      }
    }
  }
```

```
}

# --- Groupe B : correspondances à valider manuellement ---
cat("\n=== Groupe B : Correspondances à valider manuellement (score < ",
    threshold*100, "%) ===\n", sep = "")
manual_mapping <- data.frame(
  shape_commune = character(0),
  ehcvm_commune = character(0),
  score = numeric(0),
  stringsAsFactors = FALSE
)

if(length(groupB_indices) > 0) {
  for (i in groupB_indices) {
    current_shape <- shape_communes[i]
    candidate_order <- order(sim_matrix[i, ], decreasing = TRUE)
    candidates <- ehcvm_communes[candidate_order]
    candidate_scores <- sim_matrix[i, candidate_order]
    n_candidates <- min(top_n, length(candidates))
    cat(sprintf("\nPour la commune '%s' (meilleur score = %.2f),
                voici les %d meilleures correspondances :\n",
                current_shape, candidate_scores[1], n_candidates))
    for (k in 1:n_candidates) {
      cat(sprintf("  %d: '%s' (score = %.2f)\n", k, candidates[k],
                  candidate_scores[k]))
    }
    rep_choice <- as.numeric(readline(prompt = "Choisissez le numéro
                                      correspondant (ou 0 pour aucune
                                      correspondance) : "))
    if(!is.na(rep_choice) && rep_choice >= 1 && rep_choice <= n_candidates) {
      manual_mapping <- rbind(manual_mapping, data.frame(
        shape_commune = current_shape,
        ehcvm_commune = candidates[rep_choice],
        score = candidate_scores[rep_choice],
        stringsAsFactors = FALSE
      ))
    } else {
      manual_mapping <- rbind(manual_mapping, data.frame(
        shape_commune = current_shape,
        ehcvm_commune = NA,
        score = NA,
        stringsAsFactors = FALSE
      ))
      cat("Aucune correspondance validée pour cette commune.\n")
    }
  }
}

# Combiner les mappings issus des deux groupes
full_mapping <- rbind(mapping_A, manual_mapping)

cat("\nMapping final validé :\n")
print(full_mapping)
```

```
  if(nrow(full_mapping) == 0 || all(is.na(full_mapping$ehcvm_commune))) {
    cat("Aucune correspondance validée.\n")
    return(NA)
  }

  # Création de la clé de jointure dans data_SEN
  data_SEN$join_key <- sapply(data_SEN$ADM3_FR_clean, function(x) {
    idx <- which(full_mapping$shape_commune == x)
    if(length(idx) > 0) {
      return(full_mapping$ehcvm_commune[idx[1]])
    } else {
      return(NA)
    }
  })

  data_SEN_valid <- data_SEN[!is.na(data_SEN$join_key), ]
  valid_ehcvm <- full_mapping$ehcvm_commune[!is.na(full_mapping$ehcvm_commune)]
  data_to_merge <- data[data$commune_clean %in% valid_ehcvm, ]

  if(nrow(data_to_merge) == 0) {
    cat("Aucun enregistrement de la base ehcvm ne correspond aux communes
        validées.\n")
    return(NA)
  }

  # Réalisation de la fusion finale
  merged_data <- merge(
    data_to_merge, data_SEN_valid,
    by.x = "commune_clean", by.y = "join_key",
    all.x = TRUE, suffixes = c("_ehcvm", "_shp")
  )

  cat("\nFusion réalisée avec succès.\n")
  return(merged_data)
}
```

# Étape 3 : Chargement et Préparation des Données pour le Sénégal

Nous chargeons ici le shapefile et la base EHCVM pour le Sénégal et appliquons le nettoyage aux noms de communes.

```
# Chargement du shapefile pour le Sénégal
data_SEN <- st_read("data/Senegal/shapefiles/sen_admbnda_adm3_anat_20240520.shp",

# Chargement de la base EHCVM (format Stata)
data <- read_dta("data/Senegal/ehcvm/ehcvm_individu_sen2021.dta")
data <- to_factor(data)

# Application du nettoyage sur le nom des communes
data <- data %>% mutate(commune_clean = nettoyage_commune(commune))
```

```r
data_SEN <- data_SEN %>% mutate(ADM3_FR_clean = nettoyage_commune(ADM3_FR))
```

## Étape 4 : Lancement de la Fusion Interactive

Lancez la fusion interactive pour associer les communes entre la base EHCVM et le shapefile.

```r
resultat_merge <- merge_bases_commune_interactive(data_SEN, data, threshold = 0.8,

# Affichage du résultat final
cat("\nRésultat de la fusion :\n")
print(resultat_merge)
```

> **Exemple** :
> Supposons que la base EHCVM contienne une commune nommée “pikine” et que, dans le shapefile, le nom soit légèrement différent (par exemple “Pikine” ou avec des variations d’orthographe). La procédure interactive, à travers le calcul de similarité et la validation manuelle, permettra de corriger et d’associer correctement ces entrées.

# Conclusion

Nous avons présenté deux approches pour fusionner les bases de données dans le cadre de la récupération des codes admin 3 :
- Pour le **Bénin**, une méthode de nettoyage simple et de fusion directe.
- Pour le **Sénégal**, une méthode plus fine qui s’appuie sur un nettoyage approfondi, le calcul d’une similarité entre chaînes de caractères et une validation interactive.