MIXCLUSTERING\_Nour Dass HAMMADI-Farida BENCHALAL

Nour Dass HAMMADI & Farida BENCHALAL

14/03/2021

#' Mixclustering  
  
#' @param X  
#' @param clust  
#' @param iterations  
#' @param initialisation  
  
#' @return  
#' @export  
  
  
library(dplyr)

## Warning: package 'dplyr' was built under R version 3.6.3

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(tidyverse)

## Warning: package 'tidyverse' was built under R version 3.6.3

## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.3 v purrr 0.3.3  
## v tibble 3.0.3 v stringr 1.4.0  
## v tidyr 1.0.2 v forcats 0.4.0  
## v readr 1.3.1

## Warning: package 'ggplot2' was built under R version 3.6.3

## Warning: package 'tibble' was built under R version 3.6.3

## Warning: package 'readr' was built under R version 3.6.3

## -- Conflicts ------------------------------------------ tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

library(gtools)

## Warning: package 'gtools' was built under R version 3.6.3

library(FactoMineR)

## Warning: package 'FactoMineR' was built under R version 3.6.3

library(bayess)

## Warning: package 'bayess' was built under R version 3.6.3

## Loading required package: MASS

## Warning: package 'MASS' was built under R version 3.6.3

##   
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':  
##   
## select

## Loading required package: mnormt

## Loading required package: gplots

##   
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':  
##   
## lowess

## Loading required package: combinat

##   
## Attaching package: 'combinat'

## The following object is masked from 'package:utils':  
##   
## combn

##   
## Attaching package: 'bayess'

## The following object is masked from 'package:gtools':  
##   
## rdirichlet

library(mvtnorm)

## Warning: package 'mvtnorm' was built under R version 3.6.3

library(devtools)

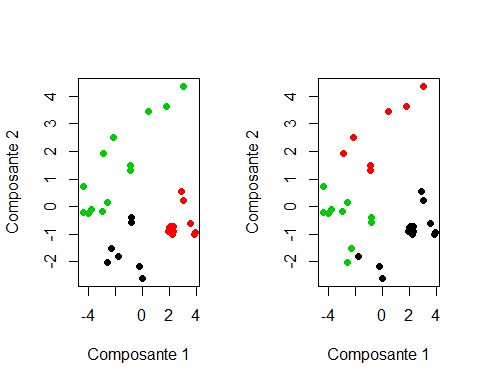
## Warning: package 'devtools' was built under R version 3.6.3

## Loading required package: usethis

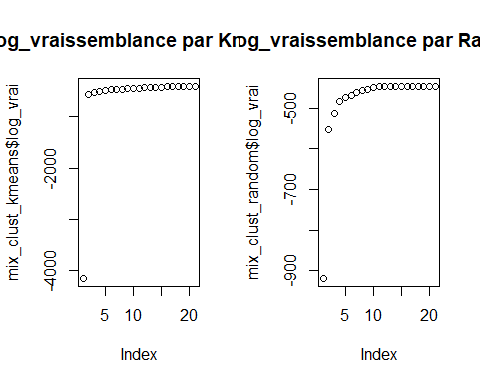
## Warning: package 'usethis' was built under R version 3.6.3

## Error in get(genname, envir = envir) : objet 'testthat\_print' introuvable

My\_Mix\_clustering <-  
 function(X, clust, iterations, initialisation) {  
 set.seed(123)  
 n <- nrow(X) # nombre de lignes  
 col <- ncol(X) # nombre de colonnes  
  
 ## S?paration des variables qualitatives et variables quantitatives:  
  
 ## Varibales quantitatives  
 donnees\_quati <- as.matrix(X %>% select\_if(is.numeric))  
 col <- ncol(donnees\_quati)  
 ## Variables qualitatives  
 donnees\_quali <- X %>% select\_if(is.factor)  
 col\_quali <- ncol(donnees\_quali)  
  
 mod <- sapply(seq(col\_quali), function(i) {  
 length(levels(donnees\_quali[, i]))  
 })  
 # initialisation des objets  
 prop <- matrix(NA, iterations + 1, clust)  
 mu <- array(NA, dim = c(iterations + 1, clust, col))  
 sigma <- array(NA, dim = c(iterations + 1, clust, col, col))  
 alpha <- array(NA, dim = c(iterations + 1, clust, col\_quali))  
 mode(alpha) <- "list"  
 log\_vrai <- rep(0, iterations + 1)  
 # initialisation de l'algorithme => random/kmeans  
 if (initialisation == 'random') {  
 prop[1,] <- rdirichlet(1, par = rep(1, clust))  
 mu[1, ,] <- donnees\_quati[sample(1:n, clust),]  
 for (k in 1:clust)  
 sigma[1, k, ,] <- rWishart(1, 8, var(donnees\_quati))  
 }  
 if (initialisation == 'kmeans') {  
 z <- kmeans(donnees\_quati, clust)$clust  
 for (k in 1:clust) {  
 prop[1, k] <- mean(z == k)  
 mu[1, k,] <- colMeans(donnees\_quati[which(z == k),])  
 sigma[1, k, ,] <- var(donnees\_quati[which(z == k),])  
 }  
 }  
 # initialisation des parametres  
 for (k in 1:clust) {  
 for (i in 1:col\_quali) {  
 alpha[1, k, i] <- list(rdirichlet(1, rep(1, mod[i])))  
 names(alpha[1, k, i][[1]]) <- levels(donnees\_quali[, i])  
 }  
 }  
 # calcul de log de vraisemblance  
 for (i in 1:n) {  
 tmp <- 0  
 for (k in 1:clust) {  
 fk <- 1  
 for (j in 1:col\_quali) {  
 fk <-fk \* alpha[1, k, j][[1]][donnees\_quali[i, j]]  
 }  
 tmp <-  
 tmp + prop[1, k] \* (fk \* dmvnorm(donnees\_quati[i,], mu[1, k,], sigma[1, k, ,]))  
 }  
 log\_vrai[1] <- log\_vrai[1] + log(tmp)  
 }  
 # algorithme EM  
 for (iter in 1:iterations) {  
 #E-step  
 tik <- matrix(NA, n, clust)  
 for (k in 1:clust) {  
 fk <- 1  
 for (i in 1:col\_quali) {  
 fk <- fk \* alpha[iter, k, i][[1]][donnees\_quali[, i]]  
 }  
 tik[, k] <-  
 prop[iter, k] \* (fk + dmvnorm(donnees\_quati, mu[iter, k,], sigma[iter, k, ,]))  
 }  
 tik <- tik / rowSums(tik)  
 #M-step  
 for (k in 1:clust) {  
 nk <- sum(tik[, k])  
 prop[iter + 1, k] <- nk / n  
 mu[iter + 1, k,] <- colSums(tik[, k] \* donnees\_quati) / nk  
 sigma[iter + 1, k, ,] <- Reduce('+', lapply(1:n, function(m) {  
 tik[m, k] \* (donnees\_quati[m,] - mu[iter + 1, k,]) %\*% t(donnees\_quati[m,] - mu[iter + 1, k,]) / nk  
 }))  
 for (i in 1:col\_quali) {  
 alpha[iter + 1, k, i] <- list(sapply(1:mod[i], function(a) {  
 sum(tik[, k] \* (donnees\_quali[, i] == levels(donnees\_quali[, i])[a])) / nk  
 }))  
 names(alpha[iter + 1, k, i][[1]]) <- levels(donnees\_quali[, i])  
 }  
 }  
 #calcul de log vraisemblance  
 for (i in 1:n) {  
 tmp <- 0  
 for (k in 1:clust) {  
 fk <- 1  
 for (j in 1:col\_quali) {  
 fk <-fk \* alpha[iter + 1, k, j][[1]][donnees\_quali[i, j]]  
 }  
 tmp <-  
 tmp + prop[iter + 1, k] \* (fk \* dmvnorm(donnees\_quati[i,], mu[iter + 1, k,], sigma[iter + 1, k, ,]))  
 }  
 log\_vrai[iter + 1] <- log\_vrai[iter + 1] + log(tmp)  
 }  
 }  
 z <- max.col(tik)  
 BIC <- log\_vrai[iterations + 1] - clust / 2 \* log(n)  
 ICL <- BIC - sum(tik \* log(tik), na.rm = TRUE)  
  
 return(  
 list(  
 prop = prop,  
 mu = mu,  
 sigma = sigma,  
 clust = clust,  
 log\_vrai = log\_vrai,  
 z = z,  
 BIC = BIC,  
 ICL = ICL  
 )  
 )  
 }  
  
  
###############################  
  
  
x <- mtcars  
x$vs = as.factor(x$vs)  
x$am = as.factor(x$am)  
x$gear = as.factor(x$gear)  
  
mix\_clust\_kmeans <- My\_Mix\_clustering(x, 3, 20, 'kmeans')  
mix\_clust\_random <- My\_Mix\_clustering(x, 3, 20, 'random')  
  
#############################  
  
  
affiche\_graphe <- function(resultat\_mixclust){  
 res.famd <- FAMD (x, ncp = 5, graph = FALSE)  
 plot(res.famd$ind$coord[,1], res.famd$ind$coord[,2],  
 # col = res$z, pch=19, xlab = "Composante 1", ylab = "Composante 2")  
 col = resultat\_mixclust$z, pch=19, xlab = "Composante 1", ylab = "Composante 2")  
}  
  
  
  
  
  
set.seed(123)  
par(mfrow = c(1,2))  
affiche\_graphe(mix\_clust\_kmeans)  
affiche\_graphe(mix\_clust\_random)



par(mfrow = c(1,2))  
plot(mix\_clust\_kmeans$log\_vrai, main = "Log\_vraissemblance par Kmeans")  
plot(mix\_clust\_random$log\_vrai, main = "Log\_vraissemblance par Random"  
 )



res.famd <- FAMD (x, ncp = 5, graph = FALSE)  
par(mfrow = c(1,2))  
plot(res.famd$ind$coord[,1], res.famd$ind$coord[,2],  
 col = mix\_clust\_kmeans$z, pch=19, xlab = "Composante 1", ylab = "Composante 2", main = "Mix Clustering par Kmeans")  
plot(res.famd$ind$coord[,1], res.famd$ind$coord[,2],  
 col = mix\_clust\_random$z, pch=19, xlab = "Composante 1", ylab = "Composante 2", main = "Mix Clustering par Random")

