Projet Kmeans ++

Louan Ourvouai – Valentin Buchon

02 Novembre 2022

Exercice 1

Question 1

```
kmeanspp <- function(X,K){
  len <- dim(X)[1]
  res <- c(sample(1:len,1))
  max <- K-1
  for (k in seq(1:max)){# Pour avoir K points
    dist_tab <- c() # Notre vecteur de proba distance
    idx <- seq(1:len)
    idx <- idx[!(seq(1:len) %in% res)]
    for (i in idx){
      dist <- .Machine$integer.max
      for(j in res){ # Pour tous les points c_i déjà sélectionner
        aux <- euclidean_dist(X[i,],X[j,])
        if (aux < dist){
          dist <- aux
        } # On prend le min de la distance
      }
      dist_tab <- append(dist_tab,dist)
    }
    dist_tab <- dist_tab^2/sum(dist_tab^2)
    # On doit choisir un point avec une proba pondéré par la distance
    c_k <- sample(idx,1,prob=dist_tab)
    res <- append(res,c_k)
  }
  res
}



kmean <- function(X,K){
  X <- as.matrix(X)
  len <- dim(X)[1]
  size <- dim(X)[2]
  C <- kmeanspp(X,K)
  centerMass <- matrix(0,nrow = K, ncol = size)
  meanMass <- matrix(0,nrow = K, ncol = size)
  for(mass in seq(1:K)){
    l <- C[mass]
    centerMass[mass,] <- X[l,]
```

```
  }
  C <- seq(1:K)
  while(euclidean_dist(centerMass,meanMass) > 0.001){
    cluster <- seq(1:len)
    for (i in seq(1:K)){ # Pour chaque i
      for (x in seq(1:len)){ # On va tester tous les x
        b <- TRUE
        loop <- seq(1:K)
        loop <- loop[!(seq(1:K) %in% i)]
        for (j in loop){ # On va faire la comparaison de la distance avec tous les autres
          if (euclidean_dist(centerMass[i,],X[x,])>euclidean_dist(centerMass[j,],X[x,])){
            b <- FALSE
          }
        }
        if(b){
          cluster[x] <- C[i]
        }
      }
    }
    meanMass <- matrix(0,nrow = K, ncol = size)
    for(i in seq(1:len)){
      for (c in C)
        if(cluster[i] == c){
          meanMass[c,] <- meanMass[c,] + X[i,]
        }
    }
    summ <- table(cluster)

    for(c in C){
      meanMass[c,] <- meanMass[c,]/summ[c]
    }

    aux <- centerMass
    centerMass <- meanMass
    meanMass <- aux

  }
  centerMass
}
```

Question 2

```
# n doit être un multiple de x
normx <- function(x,d,n){
  if (n%%x != 0){
    stop("n n est pas un multiple de x")
  }
  len <- n/x
  center <- matrix(nrow = x, ncol = d)
  res <- matrix(nrow = n, ncol = d)
  for (i in seq(1:x)){
    center[i,] <- runif(d,0,500)
    for (j in seq(1:d)){
```

```
      vec <- rnorm(len,mean=center[i,j],sd=1)
      debut <- ((i-1)*len)+1
      fin <- i*len
      res[debut:fin,j] <- vec
    }
    res[i*len,] <- center[i,]
  }
  res
}
```

Question 3

Voici nos résultats pour la comparaison avec NORM-10, NORM-25 pour kmeans et kmeans++. Nous retrouvons le même comportement avec les mêmes écarts relatifs que dans le papier de recherche. Cependant pour une raison encore incconu, nous n'obtenons pas les mêmes valeurs de ϕ que dans le papier de recherche. quand $k = 10$ pour NORM-10 et quand $k = 25$ pour NORM-25, kmeans++ est nettement plus performant, il est plus rapide et plus précis.

k	mean_phi_k	mean_phi_kpp	min_phi_k	min_phi_kpp	time_k	time_kpp
10	234.42	4.7	4.7	4.7	5.7	1.68
25	8.11	2.11	1.98	1.89	42.49	49.32
50	1.1	0.97	0.95	0.86	212.93	217.32

Tableau comparatif de kmeans++ et kmeans pour NORM-25 avec n = 1000 et d = 5

k	mean_phi_k	mean_phi_kpp	min_phi_k	min_phi_kpp	time_k	time_kpp
10	328.55	257.14	150.02	117.71	4.8	4.04
25	27.24	3.06	6.04	3.06	26.51	8.75
50	2.88	1.83	1.82	1.74	152.03	134.44

Tableau comparatif de kmeans++ et kmeans pour NORM-25 avec n = 1000 et d = 5

Exercice 2

Question 1

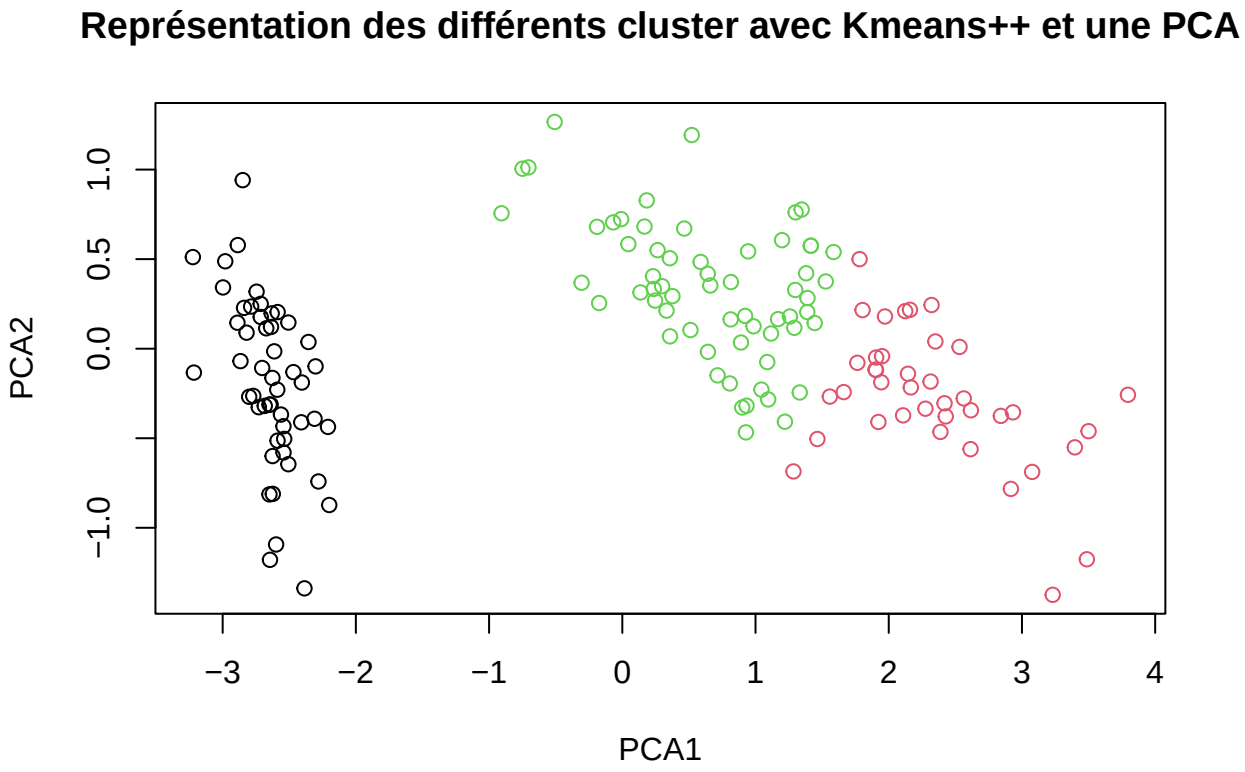
Voici le cardinal de chacun des 3 sous ensemble :

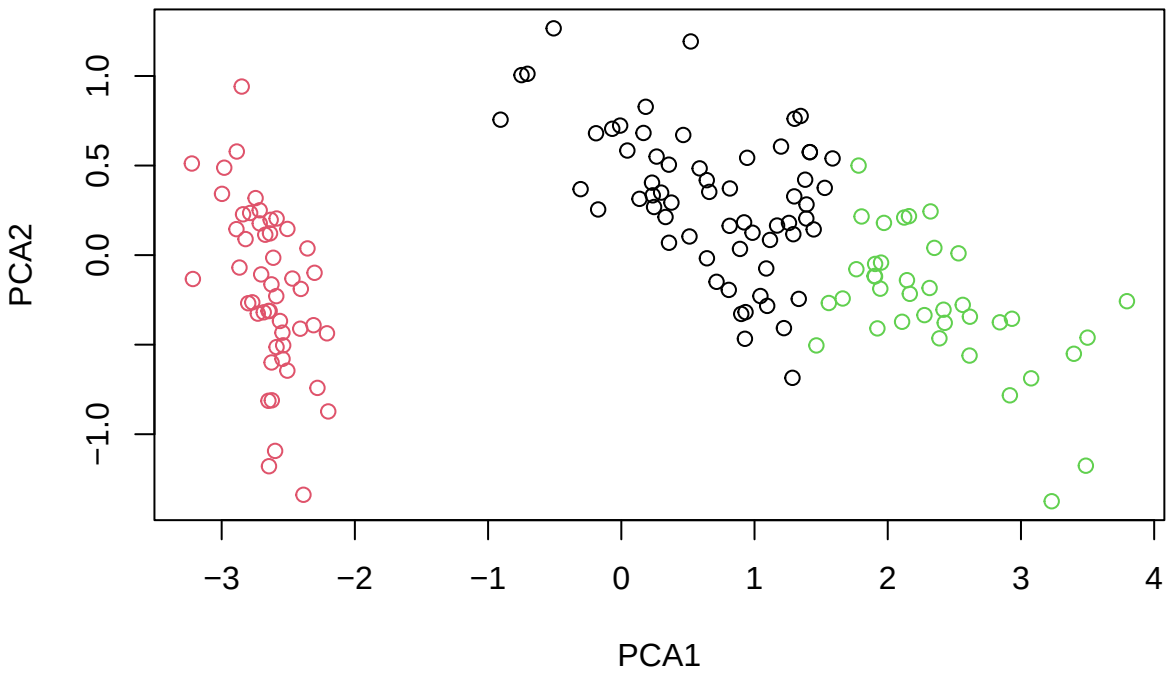
Nombre de cluster pour kmeans++ : 50 39 61

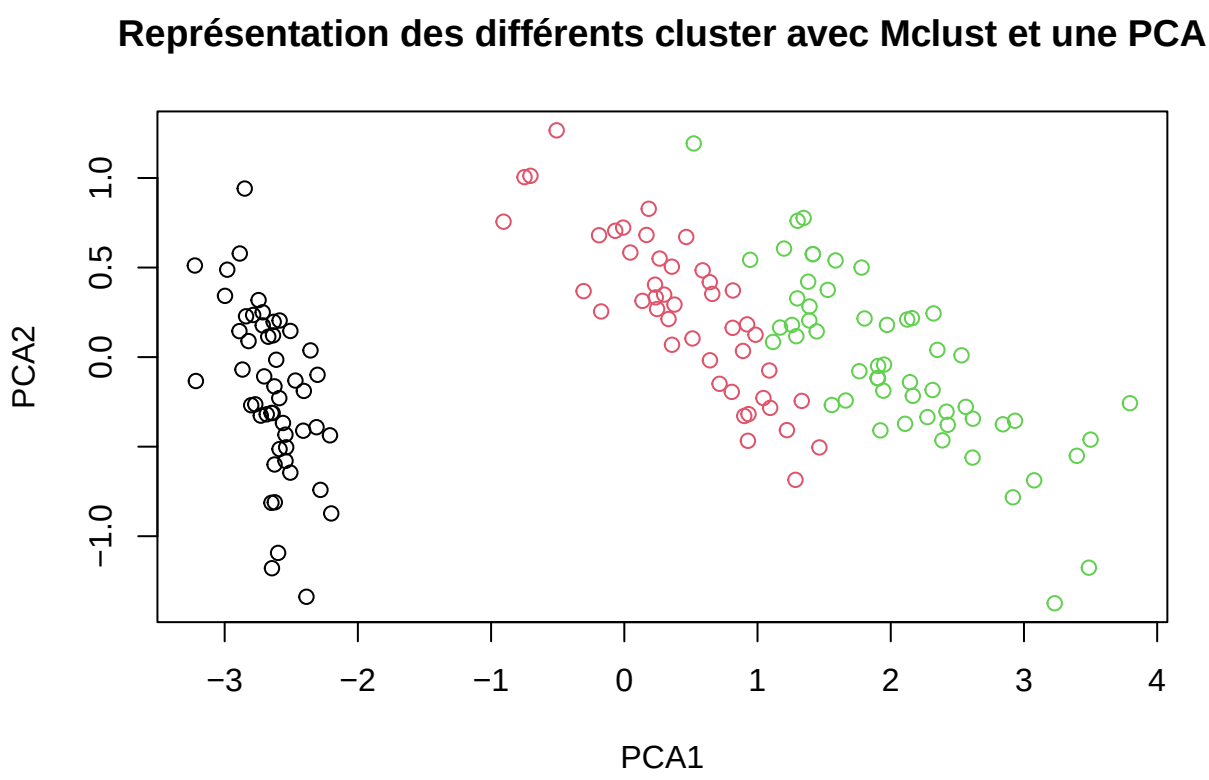
Nombre de cluster pour kmeans : 62 50 38

Nombre de cluster pour Hclust : 50 45 55

Question 2


Représentation des différents cluster avec Kmeans++ et une PCA
PCA2
1.0
0.5
0.0
−1.0
−3
−2
−1
0
1
2
3
4
PCA1


Représentation des différents cluster avec Kmeans et une PCA
PCA2
1.0
0.5
0.0
−1.0
−3
−2
−1
0
1
2
3
4
PCA1



Question 3

Kmeans++ et Kmeans donnent des résultats quasiement similaire, Mclust lui est un peu différents. La non différence de Kmeans++ et Kmeans semble logique, Kmeans++ permet de converger plus vite et déviter certaines abérations, mais sinon ce sont les mêmes algorithmes. Mclust est pas nature différents.