Détermination du nombre optimal des classes

Jguirim Nawres Alfahem Rihab Smii Faouzi

11 decembre 2017

Plan

I. Introduction

Parfois choisir le bon nombre de classe pour un modèle de classification peut être problématique, pour cela le but de notre projet est de construire une fonction qui facilite cette tâche, en retournant l'indice de rand et le meilleur nombre de classe; cette fonction prend comme paramètres la base de données, nombre de classe minimale et maximale, la méthode de bruitage, le taux et le nombre d'échantillonnage.
Dans ce rapport on va expliquer les étapes de construction de cette fonction, la tester sur un 4 jeux de données différents et interpréter les résultats.

II. jeu de données

1ér jeu de donnée "iris"

```
data("iris")
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

2 éme jeu de données simulé

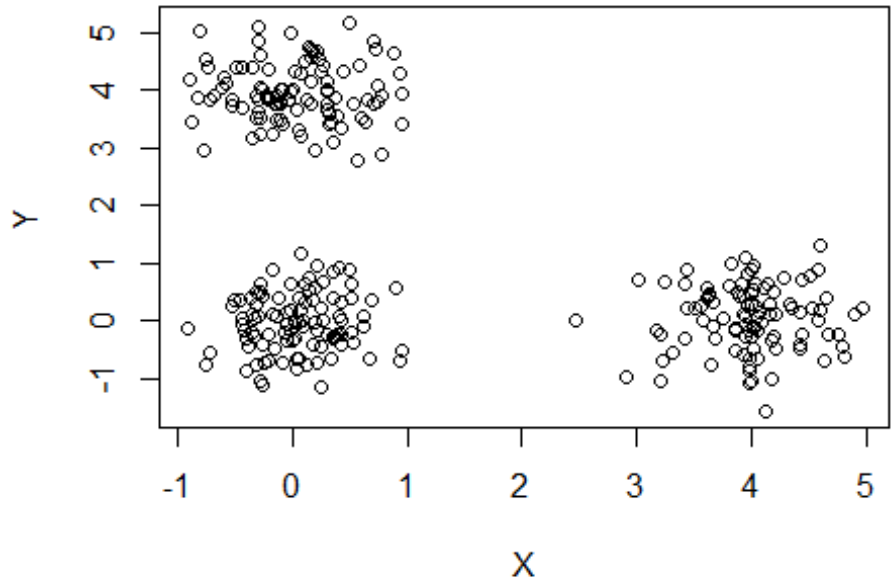
On a simulé trois bivariates distributions , ensuite on les a concaténé dans une table, et après on la projet dans une plot.

```
#simulation population 1
#function simuttion population with 3 different bivariate distribution
gaussianPop=function(N, mu1,mu2, mu3,  sigma){
  library(MASS)
  bvn1 <- mvrnorm(N, mu = mu1, sigma )
  bvn2 <- mvrnorm(N, mu = mu2, sigma  )
  bvn3 <- mvrnorm(N, mu = mu3, sigma  )
  base=rbind.data.frame(bvn1, bvn2, bvn3)
  colnames(base) <- c("X","Y")
  return(base)
}
# Parameters for bivariate normal distribution
N=100
mu1 <- c(0,0) # Mean
mu2 <- c(0,4) # Mean
```

```r
mu3 <- c(4,0) # Mean)
# Covariance matrix
sigma <- matrix(c(0.2,0,0,0.3),  2)
base1=gaussianPop(N, mu1,mu2, mu3, sigma)
head(base1)
```

```
##              X            Y
## 1  0.034332588 -0.815448674
## 2 -0.308406541 -0.755941276
## 3 -0.005402794  0.205987259
## 4  0.020138149 -0.292572104
## 5  0.392737903 -0.216880514
## 6  0.271915240 -0.002988029
```

```r
plot(base1)
```

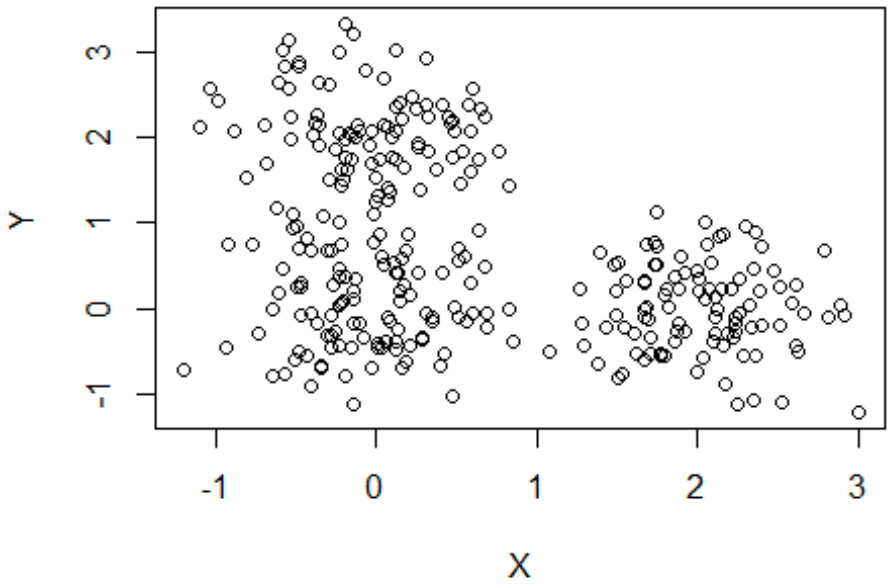


On visualise clairement qu'il ya 3 classes à détecter. Le but de ce jeu de données est de vérifier si la fonction retourne 3 comme meilleur nombre de classes.

3 éme jeu de données simulé

De même, on refait la simulation, mais cette fois en rapprochant les deux moyennes à l'origine, le but de cette distribution est de vérifié si l'indice de rand se diminuera ou pas.

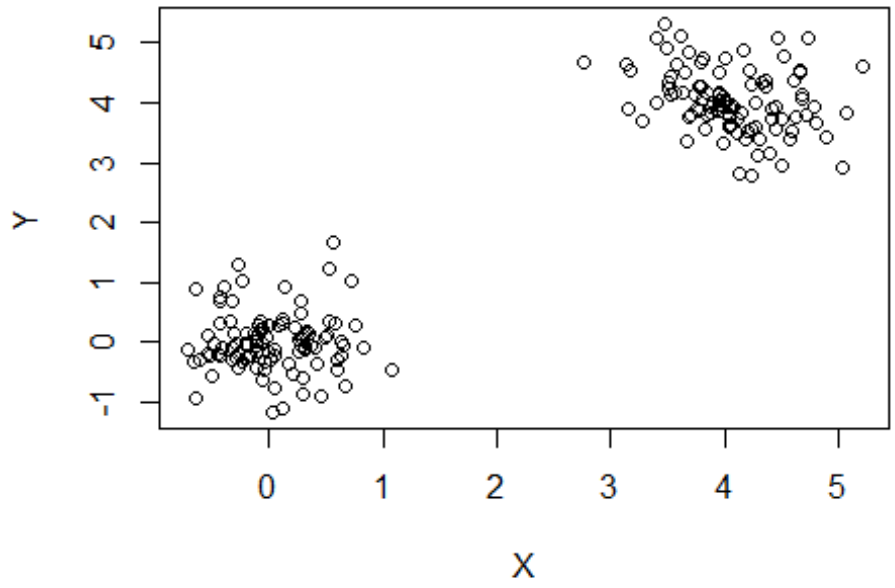
```
#simulation population 2 , on rapproche les moyennes
mu11 <- c(0,0) # Mean
mu22 <- c(0,2) # Mean
mu33<- c(2,0) # Mean
 # Covariance matrix
sigma <- matrix(c(0.2,0,0,0.3),  2)
base2=gaussianPop(N, mu11 ,mu22 , mu33,  sigma)
plot(base2)
```



4 éme jeu de données simulé

De la même manière, juste cette fois, on simule deux gaussiennes, le but de cette distribution est de vérifié si la fonction retourne 2 classes.

```
#####  simulation population deux gaussienne
mu1 <- c(0,0) # Mean
mu2 <- c(4,4) # Mean
 # Covariance matrix
sigma <- matrix(c(0.2,0,0,0.3),  2)
bvn1 <- mvrnorm(N, mu = mu1, Sigma = sigma )
bvn2 <- mvrnorm(N, mu = mu2, Sigma = sigma )
towGauss=rbind.data.frame(bvn1, bvn2)
colnames(towGauss) <- c("X","Y")
plot(towGauss)
```



III. La fonction

1. La table de classification

Le but de cette fonction est de retourner la table contenant la classification, de l'échantillon, et de la population. Elle prend comme paramètre la base de données, taux d'échantillonnage, méthode de bruitage (remise sans remise et stratification) et la méthode de classification(kmeans et hclust).
Au niveau de la méthode kmeans , on a a choisi l'algorithme par défaut Hartigan-Wong parce qu'il converge rapidement, par contre on risque de tomber dans un minimum local au lieu d'un global, d'où la nécessité d'effectuer plusieurs itérations pour atteindre une classification de bonne qualité.
ET pour l'algorithme hclust (classification hiérarchique) avec la méthode « dist » (Distance Matrix Computation)

```
clusterTable=function(x, tau, k , method  , algo){
  library(splitstackshape)
  library( dplyr)
  if (method=="sans remise")  xEch =  sample_n(x, round(nrow(x)*tau),
replace= FALSE)
  if (method=="avec remise") xEch =  sample_n(x, round(nrow(x)*tau),
replace= TRUE)
  if (algo=="kmeans") {
    res1=kmeans(x, k, iter.max = 40, nstart = 4, algorithm = "Hartigan-
Wong", trace=FALSE)
  #plot(x, col = res1$cluster)
 # points(res1$centers, pch = 8, cex = 2)
  if (method== "strat") {
    x= cbind(x,c(1:nrow(x)),res1$cluster)
    xEch=stratified(x ,colnames(x)[ ncol(x)], tau )
    nam =xEch$`c(1:nrow(x))`
    n=ncol(xEch)-2
    xEch=xEch[,c(1:n), with=FALSE]
     rownames(xEch)=NULL
    rownames(xEch)=nam
    }
  res2=kmeans(xEch, k, iter.max = 40, nstart = 4, algorithm =
"Hartigan-Wong", trace=FALSE)
  a=as.data.frame (res2$cluster)
  b=as.data.frame (res1$cluster)
  a=cbind(as.numeric((rownames(xEch))),a)
  b=cbind(as.numeric((rownames(b))),b )
  ab=data.frame(rep(0,nrow(xEch)),rep(0,nrow(xEch)))
  for (i in 1 : nrow(xEch)){
  ab[i,]=b[a[i,1],] }
  ab=cbind.data.frame(ab,a[,2])
  colnames(ab)=c( "ind","cluster pop", "cluster ech")  }

  if(algo=="hclust"){
```

```
    res1=hclust(dist(x)^2)
    c1=cutree(res1, k)
  if ( method=="strat") {
    x= cbind(x,c(1:nrow(x)),c1)
    xEch=stratified(x ,colnames(x)[ ncol(x)], tau )
    nam =xEch$`c(1:nrow(x))`
    n=ncol(xEch)-2
    xEch=xEch[,c(1:n), with=FALSE]
    rownames(xEch)=NULL
    rownames(xEch)=nam
  }
    res2=hclust(dist(xEch)^2)
    c2=cutree(res2, k)
    a=as.data.frame (c2)
    b=as.data.frame (c1)
    a=cbind(as.numeric((rownames(xEch))),a)
    b=cbind(as.numeric((rownames(b))),b )
    ab=data.frame(rep(0,nrow(xEch)),rep(0,nrow(xEch)))
    for (i in 1 : nrow(xEch)){
      ab[i,]=b[a[i,1],] }
    ab=cbind.data.frame(ab,a[,2])
    colnames(ab)=c( "ind","cluster pop", "cluster ech")
    }

  return(ab)}
```

Exemple :

```
clusterTable(iris[,-5],0.2, 4,"strat" , "hclust")
ind cluster pop cluster ech
1    3           1           1
2   34           1           1
3   42           1           1
4   23           1           1
5    9           1           1
6   25           1           1
7    4           1           1
8   27           1           1
9   18           1           1
10  11           1           1
11  55           2           2
```

2. Calcule de l'indice de rand

Dans cette fonction on a utilisé le package ClustOVar et fossil,qui nous ont fournient 3 indices de rands, l'un de d'eux est ajuster, le but c'est de faire une comparaison ces indices. La fonction nous retour la moyenne des indices
k:nombre de classes
nbEch: nombre d'échantillonnage

```
listInd= function(pop,tau , k, nbEch ,method, algo){
  index=matrix(c(0,0,0),1)
  colnames(index)=c("clustOfVar","fossil", "adj.rand.fossil")
  library(mclust)
  library(ClustOfVar)
  library(fossil)
  for (j in 1:nbEch) {
    cc=clusterTable(pop,tau , k , method, algo )
    adj.indx=adjustedRandIndex(cc[,3], cc[,2])
    indx1=rand(cc[,3], cc[,2])
    indx2=rand.index(cc[,3], cc[,2])
    index[1]= (index[1]+ indx1)
    index[2]= (index[2]+ indx2)
    index[3]=(index[3]+ adj.indx)}
  index[1]= (index[1]/nbEch)
  index[2]= (index[2]/nbEch)
  index[3]=(index[3]/nbEch)
  print("la moyenne des indices ")
  print(index)
  return(index)
}
```

4. La fonction finale Validation

```
validation=function(pop,tau , kmin , kmax ,  nbEch, method, algo){
  cat("la  methode d'echontillonage est: ", method, " de taux ",tau
,"\n", " pour kmin= ", kmin, " et kmax= ", kmax, "\n\n")
  cat("Pour K= ", kmin, "\n")
  bestIndex=listInd( pop,tau , kmin,  nbEch, method, algo)
  kList=matrix(c(kmin,kmin,kmin),1)
  for (i in (kmin+1) : kmax ){
    cat("Pour K= ", i, "\n")
    index=listInd(pop,tau , i,  nbEch, method, algo)
    if ( bestIndex[1]< index[1] ) {  bestIndex[1]=index[1]
    kList[1]=i  }
    if ( bestIndex[2]< index[2] ) { bestIndex[2]=index[2]
    kList[2]=i}
    if ( bestIndex[3]< index[3] ) { bestIndex[3]=index[3]
    kList[3]=i}
  }
```

```
  bestIndex=rbind(bestIndex ,as.integer( kList))
  rownames(bestIndex)=c("Meilleur indice ", "Meilleur nombres de
classes")
  return(bestIndex)
}
```

IV. Test

Maintenant on va tester par les 4 jeux de données pour verifié si la fonction nous retourne le meilleur nombre de classe ou pas.

1. iris

```
validation(iris[,-5], .8, 2,8, 100, "strat" ,"hclust")
```

```
## la  methode d'echontillonage est:  strat  de taux  0.8
##   pour kmin=  2  et kmax=  8
##
## Pour K=  2
```

```
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.7059768 0.8528894       0.7059768
## Pour K=  3
## [1] "la moyenne des indices "
##      clustOfVar   fossil adj.rand.fossil
## [1,]    0.79909 0.906042         0.79909
## Pour K=  4
## [1] "la moyenne des indices "
##      clustOfVar   fossil adj.rand.fossil
## [1,]    0.86202 0.942084         0.86202
## Pour K=  5
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.7059009 0.8904314       0.7059009
## Pour K=  6
## [1] "la moyenne des indices "
##      clustOfVar   fossil adj.rand.fossil
## [1,]  0.7226578 0.915479       0.7226578
## Pour K=  7
## [1] "la moyenne des indices "
##      clustOfVar   fossil adj.rand.fossil
## [1,]  0.7320793 0.925119       0.7320793
## Pour K=  8
## [1] "la moyenne des indices "
##      clustOfVar   fossil adj.rand.fossil
## [1,]  0.7136787 0.930073       0.7136787
```

```
##                               clustOfVar   fossil adj.rand.fossil
## Meilleur indice                  0.86202 0.942084         0.86202
## Meilleur nombres de classes      4.00000 4.000000         4.00000
```

```
validation(iris[,-5], .8, 2,9, 100, "avec remise" , " hclust"  )
```

```
la  methode d'echontillonage est:  avec remise  de taux  0.8
  pour kmin=  2  et kmax=  9

Pour K=  2
[1] "la moyenne des indices "
     clustOfVar    fossil adj.rand.fossil
[1,]  0.6712262 0.8357353       0.6712262
Pour K=  3
[1] "la moyenne des indices "
     clustOfVar    fossil adj.rand.fossil
[1,]  0.7604377 0.8877171       0.7604377
Pour K=  4
[1] "la moyenne des indices "
     clustOfVar    fossil adj.rand.fossil
[1,]  0.8363663 0.9311765       0.8363663
Pour K=  5
[1] "la moyenne des indices "
     clustOfVar    fossil adj.rand.fossil
[1,]  0.6484932 0.8677871       0.6484932
Pour K=  6
[1] "la moyenne des indices "
     clustOfVar    fossil adj.rand.fossil
[1,]  0.7114725 0.9089874       0.7114725
Pour K=  7
[1] "la moyenne des indices "
     clustOfVar    fossil adj.rand.fossil
[1,]  0.6864712 0.9086555       0.6864712
Pour K=  8
[1] "la moyenne des indices "
     clustOfVar    fossil adj.rand.fossil
[1,]  0.7044835 0.9228445       0.7044835
Pour K=  9
[1] "la moyenne des indices "
     clustOfVar    fossil adj.rand.fossil
[1,]  0.6840858 0.9222521       0.6840858
                            clustOfVar    fossil adj.rand.fossil
Meilleur indice              0.8363663 0.9311765       0.8363663
Meilleur nombres de classes  4.0000000 4.0000000       4.0000000
```

```
validation(iris[,-5], .8, 2,9, 100, "strat" ,"kmeans")
```

```
la  methode d'echontillonage est:  strat  de taux  0.8
  pour kmin=  2  et kmax=  9

Pour K=  2
[1] "la moyenne des indices "
```

```
     clustOfVar    fossil adj.rand.fossil
[1,]  0.9986614 0.9993361       0.9986614
Pour K=  3
[1] "la moyenne des indices "
     clustOfVar    fossil adj.rand.fossil
[1,]  0.9830088 0.9923908       0.9830088
Pour K=  4
[1] "la moyenne des indices "
     clustOfVar    fossil adj.rand.fossil
[1,]  0.9041534 0.9627115       0.9041534
Pour K=  5
[1] "la moyenne des indices "
     clustOfVar    fossil adj.rand.fossil
[1,]  0.8879353 0.9616442       0.8879353
Pour K=  6
[1] "la moyenne des indices "
     clustOfVar    fossil adj.rand.fossil
[1,]  0.7781798 0.9332312       0.7781798
Pour K=  7
[1] "la moyenne des indices "
     clustOfVar    fossil adj.rand.fossil
[1,]  0.7897607 0.9447089       0.7897607
Pour K=  8
[1] "la moyenne des indices "
     clustOfVar    fossil adj.rand.fossil
[1,]  0.7631266 0.9444071       0.7631266
Pour K=  9
[1] "la moyenne des indices "
     clustOfVar    fossil adj.rand.fossil
[1,]  0.7795799 0.9529123       0.7795799
                           clustOfVar    fossil adj.rand.fossil
Meilleur indice             0.9986614 0.9993361       0.9986614
Meilleur nombres de classes 2.0000000 2.0000000       2.0000000
validation(iris[,-5], .8, 9,9, 100, "avec remise" ,"kmeans")

la  methode d'echontillonage est:  avec remise  de taux  0.8
  pour kmin=  2  et kmax=  9

Pour K=  2
[1] "la moyenne des indices "
     clustOfVar    fossil adj.rand.fossil
[1,]  0.9889198 0.9945392       0.9889198
Pour K=  3
[1] "la moyenne des indices "
     clustOfVar    fossil adj.rand.fossil
[1,]  0.9370378 0.971493       0.9370378
Pour K=  4
[1] "la moyenne des indices "
     clustOfVar    fossil adj.rand.fossil
[1,]  0.8602768 0.9448347       0.8602768
Pour K=  5
[1] "la moyenne des indices "
     clustOfVar    fossil adj.rand.fossil
[1,]  0.7930563 0.9280588       0.7930563
Pour K=  6
[1] "la moyenne des indices "
     clustOfVar    fossil adj.rand.fossil
[1,]  0.7559573 0.925028       0.7559573
```

```
Pour K=  7
[1] "la moyenne des indices "
     clustOfVar    fossil adj.rand.fossil
[1,]  0.7378823 0.9276919       0.7378823
Pour K=  8
[1] "la moyenne des indices "
     clustOfVar    fossil adj.rand.fossil
[1,]  0.7223478 0.9312969       0.7223478
Pour K=  9
[1] "la moyenne des indices "
     clustOfVar    fossil adj.rand.fossil
[1,]   0.704878 0.9344258        0.704878
                           clustOfVar    fossil adj.rand.fossil
Meilleur indice             0.9889198 0.9945392       0.9889198
Meilleur nombres de classes 2.0000000 2.0000000       2.0000000


validation(iris[,-5], .8, 9,9, 100, "sans remise" ,"kmeans")

la  methode d'echontillonage est:  sans remise  de taux  0.8
  pour kmin=  2  et kmax=  9

Pour K=  2
[1] "la moyenne des indices "
     clustOfVar    fossil adj.rand.fossil
[1,]  0.9970039 0.9985112       0.9970039
Pour K=  3
[1] "la moyenne des indices "
     clustOfVar    fossil adj.rand.fossil
[1,]  0.9703689 0.9864902       0.9703689
Pour K=  4
[1] "la moyenne des indices "
     clustOfVar    fossil adj.rand.fossil
[1,]  0.8981475 0.9601821       0.8981475
Pour K=  5
[1] "la moyenne des indices "
     clustOfVar    fossil adj.rand.fossil
[1,]  0.8714936 0.9557969       0.8714936
Pour K=  6
[1] "la moyenne des indices "
     clustOfVar    fossil adj.rand.fossil
[1,]   0.772617 0.9310532        0.772617
Pour K=  7
[1] "la moyenne des indices "
     clustOfVar   fossil adj.rand.fossil
[1,]  0.7644786 0.937444       0.7644786
Pour K=  8
[1] "la moyenne des indices "
     clustOfVar    fossil adj.rand.fossil
[1,]  0.7684056 0.9449482       0.7684056
Pour K=  9
[1] "la moyenne des indices "
     clustOfVar    fossil adj.rand.fossil
[1,]  0.7542831 0.9469608       0.7542831
                           clustOfVar    fossil adj.rand.fossil
Meilleur indice             0.8622908 0.9425112       0.8622908
Meilleur nombres de classes 4.0000000 4.0000000       4.0000000
```

Interprétation:
On remarque la méthode kmeans donne un résultat différent de hclust , ce dernier retourne 4 nombre de classes mais kmeans retourne 2 (pour les trois échantillonnages . et si on visualise les deux figure (1et 2) ,on peut constater que la classification avec 2 est meilleur que celle avec 4 . Alors dans ce exemple kmean donne meilleur résultat .

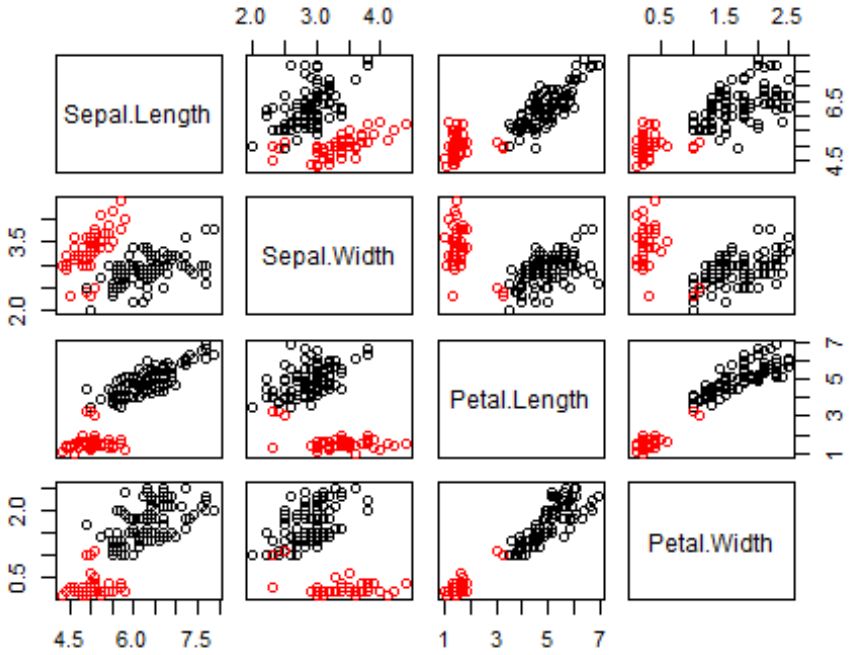


Figure 1 Visualisation d'une classification de k= 2

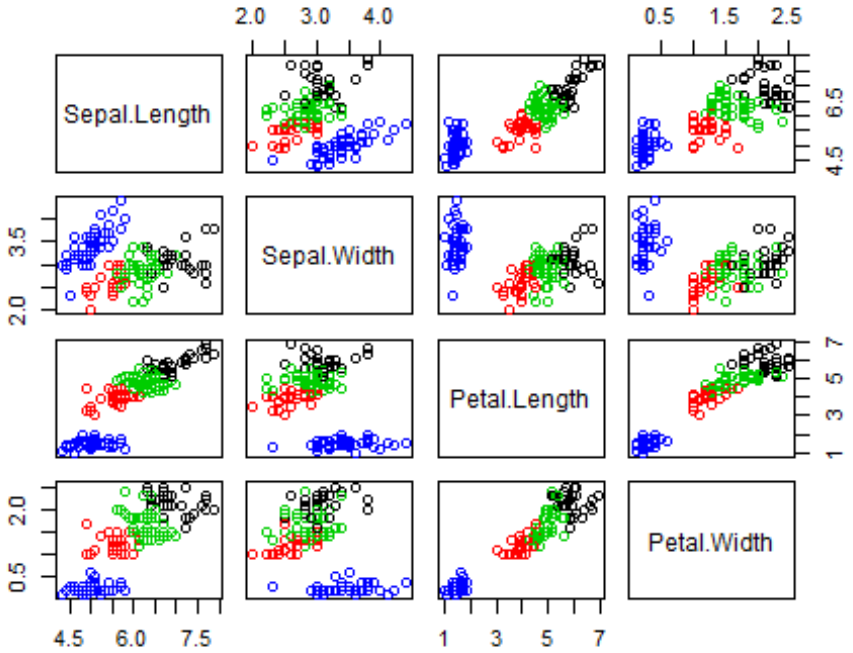


Figure 2 Visualisation d'une classification de k= 4

2. base 1

```
validation(base1, .8, 2,9, 100, "strat" ,"hclust")

## la  methode d'echontillonage est:  strat  de taux  0.8
##   pour kmin=  2  et kmax=  9
##
## Pour K=  2
## [1] "la moyenne des indices "
##      clustOfVar fossil adj.rand.fossil
## [1,]          1      1               1
## Pour K=  3
## [1] "la moyenne des indices "
##      clustOfVar fossil adj.rand.fossil
## [1,]          1      1               1
## Pour K=  4
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.9046934 0.9611433       0.9046934
## Pour K=  5
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]   0.795116 0.9262075        0.795116
## Pour K=  6
## [1] "la moyenne des indices "
```

```
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.7152643 0.9054177       0.7152643
## Pour K=  7
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.6367757 0.8955795       0.6367757
## Pour K=  8
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.5948771 0.8915642       0.5948771
## Pour K=  9
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.6137958 0.9018856       0.6137958

##                             clustOfVar fossil adj.rand.fossil
## Meilleur indice                      1      1               1
## Meilleur nombres de classes          2      2               2
```

```
validation(base1, .8, 2,9, 100, "strat" ,"kmeans")
```

```
## la  methode d'echontillonage est:  strat  de taux  0.8
##   pour kmin=  2  et kmax=  9
##
## Pour K=  2
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.7420795 0.8723776       0.7420795
## Pour K=  3
## [1] "la moyenne des indices "
##      clustOfVar fossil adj.rand.fossil
## [1,]          1      1               1
## Pour K=  4
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.8936604 0.9575404       0.8936604
## Pour K=  5
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.7802976 0.9239695       0.7802976
## Pour K=  6
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.7326334 0.9186416       0.7326334
## Pour K=  7
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.7125885 0.9223819       0.7125885
## Pour K=  8
## [1] "la moyenne des indices "
```

```
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.6746867 0.9228561       0.6746867
## Pour K=  9
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.6641127 0.9284001       0.6641127

##                             clustOfVar fossil adj.rand.fossil
## Meilleur indice                      1      1               1
## Meilleur nombres de classes          3      3               3
```

```
validation(base1, .8, 2,9, 100, "sans remise" ,"kmeans")
```

```
## la  methode d'echontillonage est:  sans remise  de taux  0.8
##   pour kmin=  2  et kmax=  9
##
## Pour K=  2
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.7383321 0.8706649       0.7383321
## Pour K=  3
## [1] "la moyenne des indices "
##      clustOfVar fossil adj.rand.fossil
## [1,]          1      1               1
## Pour K=  4
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.9244188 0.9698267       0.9244188
## Pour K=  5
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.7763626 0.9223326       0.7763626
## Pour K=  6
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.7186263 0.9143508       0.7186263
## Pour K=  7
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.6977237 0.9190645       0.6977237
## Pour K=  8
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.7090863 0.9314282       0.7090863
## Pour K=  9
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.6406842 0.9236227       0.6406842
```

```
##                              clustOfVar fossil adj.rand.fossil
## Meilleur indice                       1      1               1
## Meilleur nombres de classes           3      3               3
```

```
validation(base1, .8, 2,9, 100, "avec remise" ,"kmeans")
```

```
## la  methode d'echontillonage est:  avec remise  de taux  0.8
##   pour kmin=  2  et kmax=  9
##
## Pour K=  2
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.6231784 0.8137545       0.6231784
## Pour K=  3
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.9944863 0.9972162       0.9944863
## Pour K=  4
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.8632976 0.9452197       0.8632976
## Pour K=  5
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.7331701 0.9063968       0.7331701
## Pour K=  6
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.6926214 0.9048351       0.6926214
## Pour K=  7
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.6752401 0.9110174       0.6752401
## Pour K=  8
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.6625345 0.9182657       0.6625345
## Pour K=  9
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.6218455 0.9176848       0.6218455
```

```
##                              clustOfVar    fossil adj.rand.fossil
## Meilleur indice               0.9944863 0.9972162       0.9944863
## Meilleur nombres de classes   3.0000000 3.0000000       3.0000000
```

Interprétation: Pour ce jeu de données artificiel, la méthode hclust donne 2 et 3 comme nombre de classe , et kmeans retourne 3 avec ces trois méthodes de bruitages, on sait bien aussi que le meilleur nombre de classe est 3 , alors encore une fois kmeans est plus robuste que hclust .

3. base 2

```
validation(base2, .8, 2,9, 100, "strat" ,"hclust")
```

```
## la  methode d'echontillonage est:  strat  de taux  0.8
##   pour kmin=  2  et kmax=  9
##
## Pour K=  2
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.7027349 0.8538096       0.7027349
## Pour K=  3
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.8998736 0.9550837       0.8998736
## Pour K=  4
## [1] "la moyenne des indices "
##      clustOfVar   fossil adj.rand.fossil
## [1,]  0.8106771 0.920772       0.8106771
## Pour K=  5
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.7494956 0.9067601       0.7494956
## Pour K=  6
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.6400184 0.8821091       0.6400184
## Pour K=  7
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.6453586 0.8995314       0.6453586
## Pour K=  8
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.6375559 0.9084376       0.6375559
## Pour K=  9
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.6114091 0.9083115       0.6114091

##                             clustOfVar    fossil adj.rand.fossil
## Meilleur indice              0.8998736 0.9550837       0.8998736
## Meilleur nombres de classes  3.0000000 3.0000000       3.0000000
```

```
validation(base2, .8, 2,9, 100, "strat" ,"kmeans")
```

```
## la  methode d'echontillonage est:  strat  de taux  0.8
##   pour kmin=  2  et kmax=  9
##
## Pour K=  2
## [1] "la moyenne des indices "
```

```
##      clustOfVar    fossil adj.rand.fossil
## [1,]    0.66408 0.8327225         0.66408
## Pour K=  3
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.9976273 0.9989494       0.9976273
## Pour K=  4
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.9073351 0.9637991       0.9073351
## Pour K=  5
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.8191472 0.9387483       0.8191472
## Pour K=  6
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.8373503 0.9520177       0.8373503
## Pour K=  7
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.8015057 0.9481602       0.8015057
## Pour K=  8
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.7250821 0.9362242       0.7250821
## Pour K=  9
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.7044628 0.9386976       0.7044628

##                              clustOfVar    fossil adj.rand.fossil
## Meilleur indice               0.9976273 0.9989494       0.9976273
## Meilleur nombres de classes  3.0000000 3.0000000       3.0000000

validation(base2, .8, 2,9, 100, "sans remise" ,"kmeans")

## la  methode d'echontillonage est:  sans remise  de taux  0.8
##   pour kmin=  2  et kmax=  9
##
## Pour K=  2
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]   0.676454 0.8389317        0.676454
## Pour K=  3
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]   0.998626 0.9993912        0.998626
## Pour K=  4
## [1] "la moyenne des indices "
```

```
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.9128168 0.9660377       0.9128168
## Pour K=  5
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.8146681 0.9374209       0.8146681
## Pour K=  6
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.8271048 0.9488954       0.8271048
## Pour K=  7
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.7945757 0.9463316       0.7945757
## Pour K=  8
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.7213026 0.9353243       0.7213026
## Pour K=  9
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.7185815 0.9410987       0.7185815

##                              clustOfVar    fossil adj.rand.fossil
## Meilleur indice                0.998626 0.9993912        0.998626
## Meilleur nombres de classes    3.000000 3.0000000        3.000000

validation(base2, .8, 2,9, 100, "avec remise" ,"kmeans")

## la  methode d'echontillonage est:  avec remise  de taux  0.8
##   pour kmin=  2  et kmax=  9
##
## Pour K=  2
## [1] "la moyenne des indices "
##      clustOfVar   fossil adj.rand.fossil
## [1,]  0.6458463 0.823698       0.6458463
## Pour K=  3
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]     0.9925 0.9966653          0.9925
## Pour K=  4
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.8413108 0.9376039       0.8413108
## Pour K=  5
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.7724568 0.9227333       0.7724568
## Pour K=  6
## [1] "la moyenne des indices "
```

```
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.7514892 0.9256018       0.7514892
## Pour K=  7
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.7349379 0.9296238       0.7349379
## Pour K=  8
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.6944369 0.9279948       0.6944369
## Pour K=  9
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.6545688 0.9263033       0.6545688

##                              clustOfVar    fossil adj.rand.fossil
## Meilleur indice                  0.9925 0.9966653          0.9925
## Meilleur nombres de classes      3.0000 3.0000000          3.0000
```

Interpretation: Pour ce jeu de données , on a rapproché les moyennes de chaque Gaussienne , pour voir si les valeur des indices diminuent. Effectivelent, globalement les valeurs ont diminués.
On remarque aussi, que le nombre de classe du hclust passe de 2 à 3, et pour kmeans reste le même .

4. towGauss

```
validation(towGauss, .8, 2,9, 100, "strat" ,"hclust")

## la  methode d'echontillonage est:  strat  de taux  0.8
##   pour kmin=  2  et kmax=  9
##
## Pour K=  2
## [1] "la moyenne des indices "
##      clustOfVar fossil adj.rand.fossil
## [1,]          1      1               1
## Pour K=  3
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.7908646 0.8995362       0.7908646
## Pour K=  4
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.6959955 0.8708687       0.6959955
## Pour K=  5
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.7112296 0.8826087       0.7112296
## Pour K=  6
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
```

```
## [1,]  0.6336046 0.8696336       0.6336046
## Pour K=  7
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.6301802 0.8884143       0.6301802
## Pour K=  8
## [1] "la moyenne des indices "
##      clustOfVar   fossil adj.rand.fossil
## [1,]  0.6565318 0.904635       0.6565318
## Pour K=  9
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.6368896 0.9032426       0.6368896

##                              clustOfVar fossil adj.rand.fossil
## Meilleur indice                       1      1               1
## Meilleur nombres de classes           2      2               2

validation(towGauss, .8, 2,9, 100, "strat" ,"kmeans")

## la  methode d'echontillonage est:  strat  de taux  0.8
##   pour kmin=  2  et kmax=  9
##
## Pour K=  2
## [1] "la moyenne des indices "
##      clustOfVar fossil adj.rand.fossil
## [1,]          1      1               1
## Pour K=  3
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.8714816 0.9396085       0.8714816
## Pour K=  4
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.6951188 0.8777656       0.6951188
## Pour K=  5
## [1] "la moyenne des indices "
##      clustOfVar   fossil adj.rand.fossil
## [1,]   0.665939 0.883223        0.665939
## Pour K=  6
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.6547323 0.8933607       0.6547323
## Pour K=  7
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.7191984 0.9220395       0.7191984
## Pour K=  8
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
```

```
## [1,]    0.75823 0.9394044         0.75823
## Pour K=  9
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]   0.736368 0.9399477        0.736368

##                             clustOfVar fossil adj.rand.fossil
## Meilleur indice                      1      1               1
## Meilleur nombres de classes          2      2               2
```

```
validation(towGauss, .8, 2,9, 100, "sans remise" ,"kmeans")
```

```
## la  methode d'echontillonage est:  sans remise  de taux  0.8
##   pour kmin=  2  et kmax=  9
##
## Pour K=  2
## [1] "la moyenne des indices "
##      clustOfVar fossil adj.rand.fossil
## [1,]          1      1               1
## Pour K=  3
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.8734319 0.9403711       0.8734319
## Pour K=  4
## [1] "la moyenne des indices "
##      clustOfVar   fossil adj.rand.fossil
## [1,]  0.6866963 0.874408       0.6866963
## Pour K=  5
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.6286635 0.8698514       0.6286635
## Pour K=  6
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.6443909 0.8895747       0.6443909
## Pour K=  7
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.7192265 0.9210181       0.7192265
## Pour K=  8
## [1] "la moyenne des indices "
##      clustOfVar  fossil adj.rand.fossil
## [1,]  0.7439822 0.93518       0.7439822
## Pour K=  9
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.7212737 0.9358003       0.7212737

##                             clustOfVar fossil adj.rand.fossil
## Meilleur indice                      1      1               1
## Meilleur nombres de classes          2      2               2
```

```
validation(towGauss, .8, 2,9, 100, "avec remise" ,"kmeans")
```

```
## la  methode d'echontillonage est:  avec remise  de taux  0.8
##   pour kmin=  2  et kmax=  9
##
## Pour K=  2
## [1] "la moyenne des indices "
##      clustOfVar fossil adj.rand.fossil
## [1,]          1      1               1
## Pour K=  3
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.7971723 0.9041038       0.7971723
## Pour K=  4
## [1] "la moyenne des indices "
##      clustOfVar   fossil adj.rand.fossil
## [1,]  0.6367271 0.852548       0.6367271
## Pour K=  5
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.6144564 0.8630259       0.6144564
## Pour K=  6
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.6472772 0.8895755       0.6472772
## Pour K=  7
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.6666408 0.9055181       0.6666408
## Pour K=  8
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]  0.6910951 0.9209709       0.6910951
## Pour K=  9
## [1] "la moyenne des indices "
##      clustOfVar    fossil adj.rand.fossil
## [1,]     0.6937 0.9274041          0.6937

##                              clustOfVar fossil adj.rand.fossil
## Meilleur indice                       1      1               1
## Meilleur nombres de classes           2      2               2
```

Interprétation: dans cette partie tout les méthodes confirme le nombre de classe 2 ce qui confirme que la fonction “validation” donne le bon nombre de classe

VI. Conclusion

Globalement , la fonction validation retourne le nombre de classe le plus approprié , sauf pour la méthode hclust , elle peut être instable .
On remarque les méthodes de bruitage se diffèrent légèrement .
Aussi on peut noter que l'indice de rand fossil ajusté et l'indice du package clasOfVar pour ces 4 table donnent presque la même valeur, mais l'indice fossil se diffère d'eux, il a tendance à gonfler le résultat pour un mauvais classement, on peut être expliqué ca par le fait que ce package (fossil) est dédié au analyse des base des données écologique et géographique.