Kernel PCA

Bealy MECH

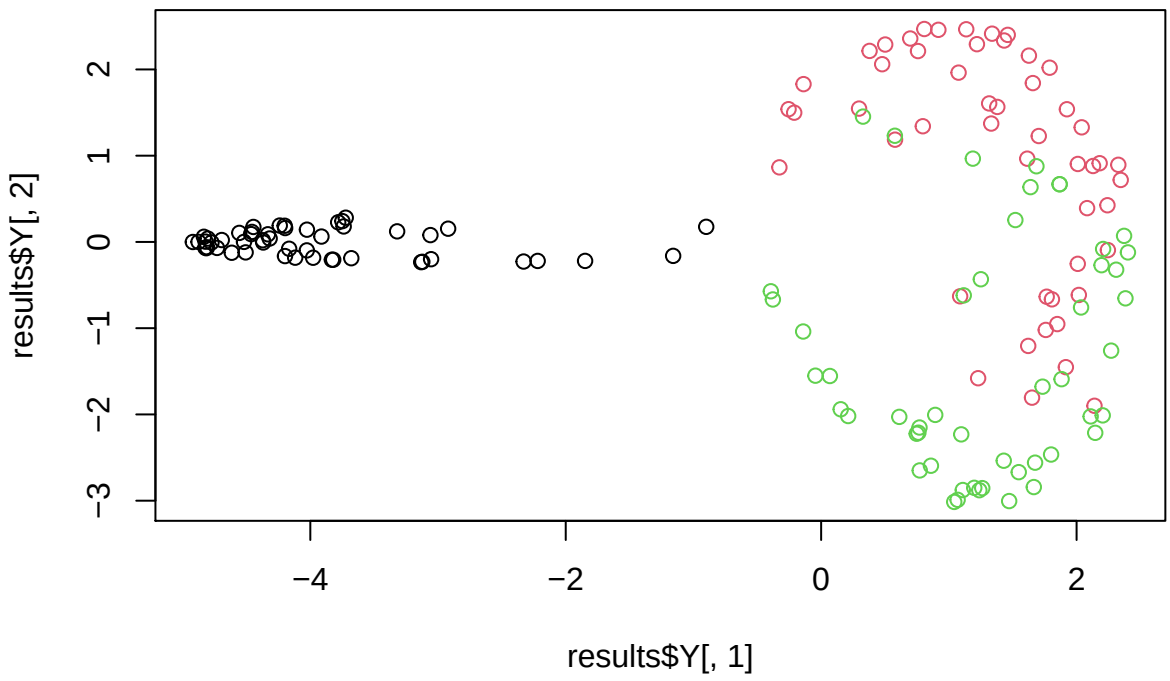
12/4/2021

Ecercise 1 : Kernel Principal Component Analysis

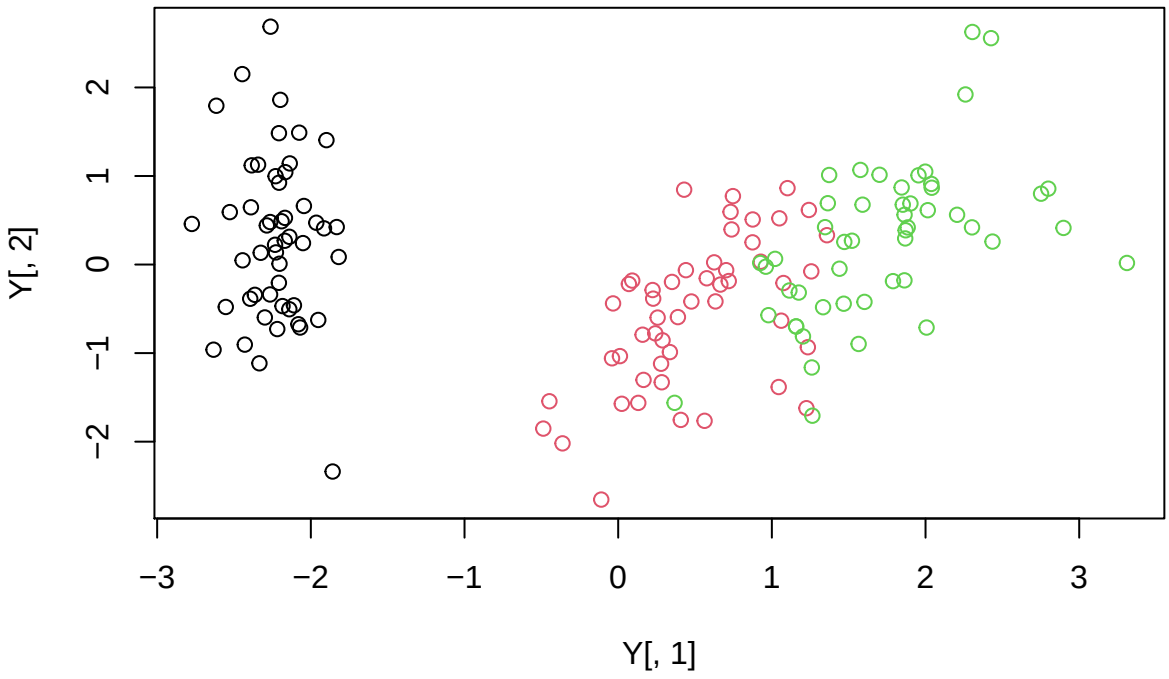
```
myKPCA <- function(X, k=2, kernel="Gaussien", beta=1){
  if (kernel == "Gaussien"){
    K <- exp(-1/beta*(as.matrix(dist(X))^2))
  } else {
    K <- X%*%t(X)
  }

  # Centering
  n <- nrow(X)
  II <- matrix(1/n,n,n)
  Ktilde <- K - 2*II%*%K + II%*%K%*%II

  # Eigenvalue decomposition
  res <- eigen(Ktilde)
  alpha <- res$vectors
  lambda <- res$values

  # Projection
  Y <- K%*%alpha[,1:k]
  return(list(Y=Y, lambda=lambda[1:k]))
}
```

```
myKPCA<-function(X,k=2,kernel="Gaussian",beta=1){
  X<-as.matrix(X)
  if (kernel == "Gaussian")
      {K<- exp(-1/beta*(as.matrix(dist(X))^2))
  } else {
    K<-X%*%t(X)
    }

  # Centering
  n<-nrow(X)
  II<-matrix(1/n,n,n)
  Ktilde<-K -2*II %*% K+ II %*%K%*%II

  # Eigenvalue decomposition
  res<-svd(Ktilde)
  alpha<-res$u
```

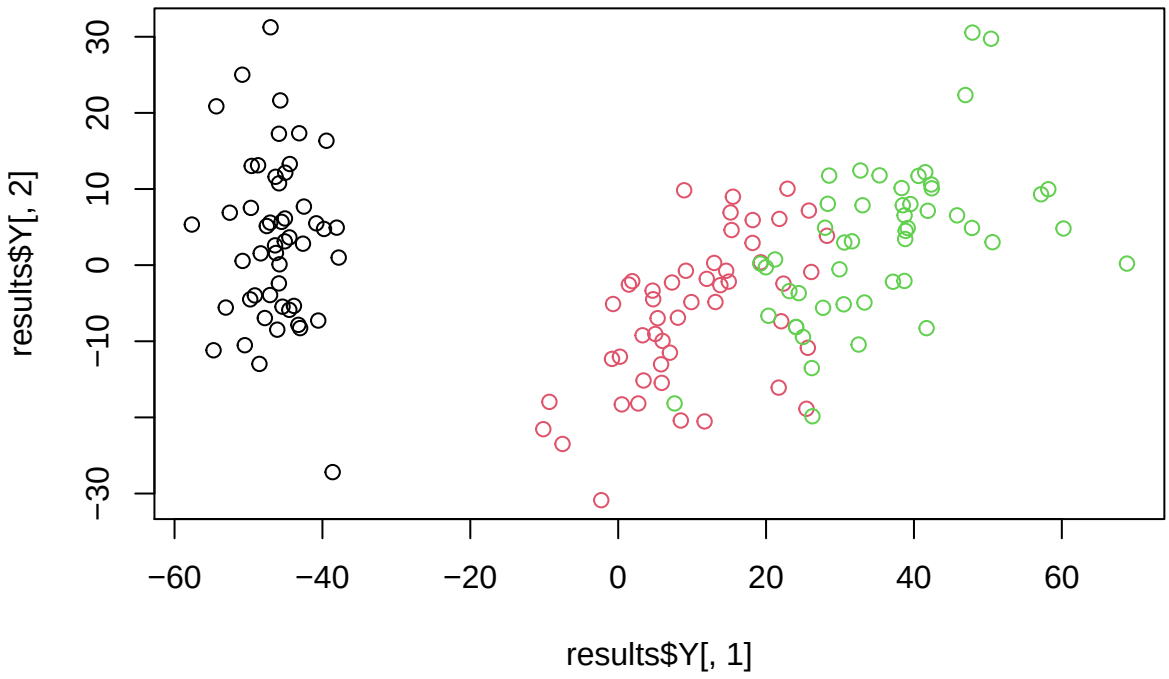
```
  lambda<-res$d^2

  # Projection
  Y<-Ktilde%*%alpha[,1:k]
  return(list(Y=Y,lambda=lambda[1:k]))
}

data(iris)
X<-iris[,1:4]

# KPCA
myKPCA(scale(X,center=TRUE,scale = TRUE),beta=2)->results
plot(results$Y[,1],results$Y[,2],col=iris$Species)
```




```
# PCA
Y<-princomp(X,cor = TRUE)$scores[,1:2]
plot(Y[,1],Y[,2],col=iris$Species)
```




```
# KPCA with linear kernel
myKPCA(scale(X,center=TRUE,scale = TRUE),kernel="Linear")->results
plot(results$Y[,1],results$Y[,2],col=iris$Species)
```




```
library(kernlab)
data(spam)
dim(spam)
```

```
## [1] 4601   58
```

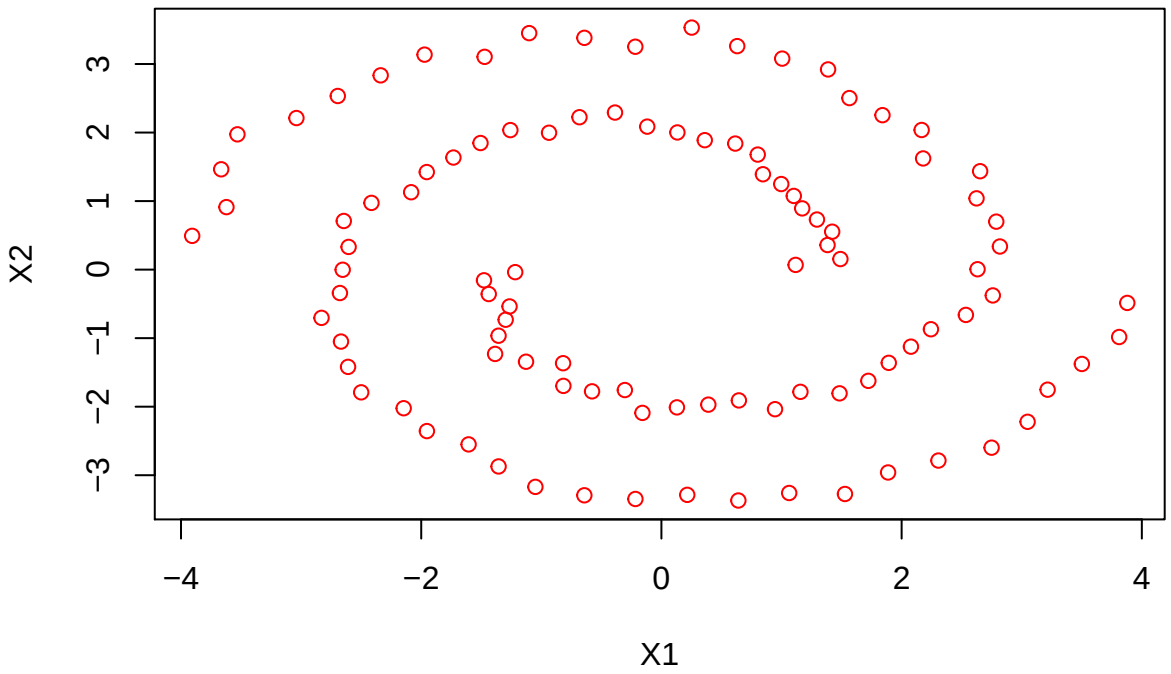
```
head(spam)
```

```
##   make address  all num3d  our over remove internet order mail receive will
## 1 0.00    0.64 0.64     0 0.32 0.00   0.00     0.00  0.00 0.00    0.00 0.64
## 2 0.21    0.28 0.50     0 0.14 0.28   0.21     0.07  0.00 0.94    0.21 0.79
## 3 0.06    0.00 0.71     0 1.23 0.19   0.19     0.12  0.64 0.25    0.38 0.45
## 4 0.00    0.00 0.00     0 0.63 0.00   0.31     0.63  0.31 0.63    0.31 0.31
## 5 0.00    0.00 0.00     0 0.63 0.00   0.31     0.63  0.31 0.63    0.31 0.31
## 6 0.00    0.00 0.00     0 1.85 0.00   0.00     1.85  0.00 0.00    0.00 0.00
##   people report addresses free business email  you credit your font num000
## 1   0.00   0.00      0.00 0.32     0.00  1.29 1.93   0.00 0.96    0   0.00
## 2   0.65   0.21      0.14 0.14     0.07  0.28 3.47   0.00 1.59    0   0.43
## 3   0.12   0.00      1.75 0.06     0.06  1.03 1.36   0.32 0.51    0   1.16
## 4   0.31   0.00      0.00 0.31     0.00  0.00 3.18   0.00 0.31    0   0.00
## 5   0.31   0.00      0.00 0.31     0.00  0.00 3.18   0.00 0.31    0   0.00
## 6   0.00   0.00      0.00 0.00     0.00  0.00 0.00   0.00 0.00    0   0.00
##   money hp hpl george num650 lab labs telnet num857 data num415 num85
## 1  0.00  0   0      0      0   0    0      0      0    0      0     0
## 2  0.43  0   0      0      0   0    0      0      0    0      0     0
## 3  0.06  0   0      0      0   0    0      0      0    0      0     0
## 4  0.00  0   0      0      0   0    0      0      0    0      0     0
## 5  0.00  0   0      0      0   0    0      0      0    0      0     0
## 6  0.00  0   0      0      0   0    0      0      0    0      0     0
##   technology num1999 parts pm direct cs meeting original project   re  edu
## 1          0    0.00     0  0   0.00  0       0     0.00       0 0.00 0.00
## 2          0    0.07     0  0   0.00  0       0     0.00       0 0.00 0.00
## 3          0    0.00     0  0   0.06  0       0     0.12       0 0.06 0.06
## 4          0    0.00     0  0   0.00  0       0     0.00       0 0.00 0.00
## 5          0    0.00     0  0   0.00  0       0     0.00       0 0.00 0.00
## 6          0    0.00     0  0   0.00  0       0     0.00       0 0.00 0.00
##   table conference charSemicolon charRoundbracket charSquarebracket
## 1     0          0          0.00           0.000                 0
## 2     0          0          0.00           0.132                 0
## 3     0          0          0.01           0.143                 0
## 4     0          0          0.00           0.137                 0
## 5     0          0          0.00           0.135                 0
## 6     0          0          0.00           0.223                 0
##   charExclamation charDollar charHash capitalAve capitalLong capitalTotal type
## 1           0.778      0.000    0.000      3.756          61          278 spam
## 2           0.372      0.180    0.048      5.114         101         1028 spam
## 3           0.276      0.184    0.010      9.821         485         2259 spam
## 4           0.137      0.000    0.000      3.537          40          191 spam
## 5           0.135      0.000    0.000      3.537          40          191 spam
## 6           0.000      0.000    0.000      3.000          15           54 spam
```

Exercise 2 : Spectral Clustering

```
library(mlbench)

set.seed(111)
```

```
obj <- mlbench.spirals(100,1,0.025)
myData <- data.frame(4 * obj$x)
names(myData) <- c("X1","X2")
plot(myData, col = "red")
```



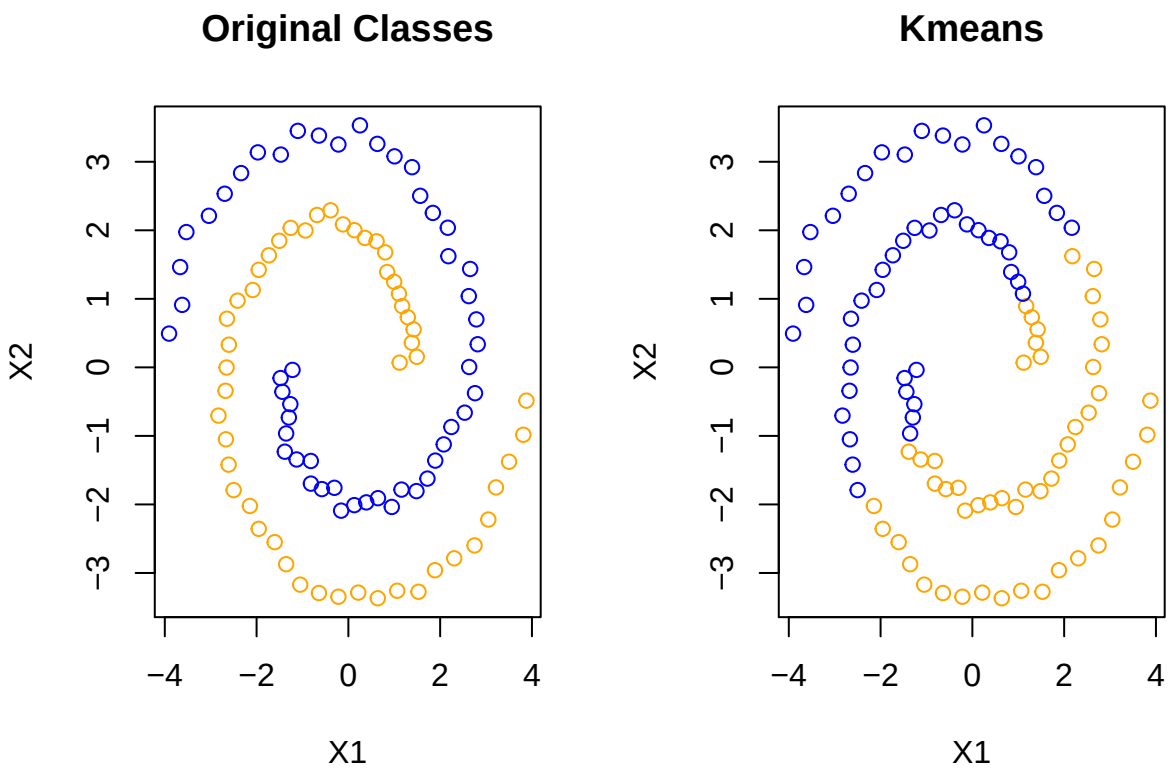

```
myData <- as.matrix(myData)
dim(myData)
```

```
## [1] 100   2
```

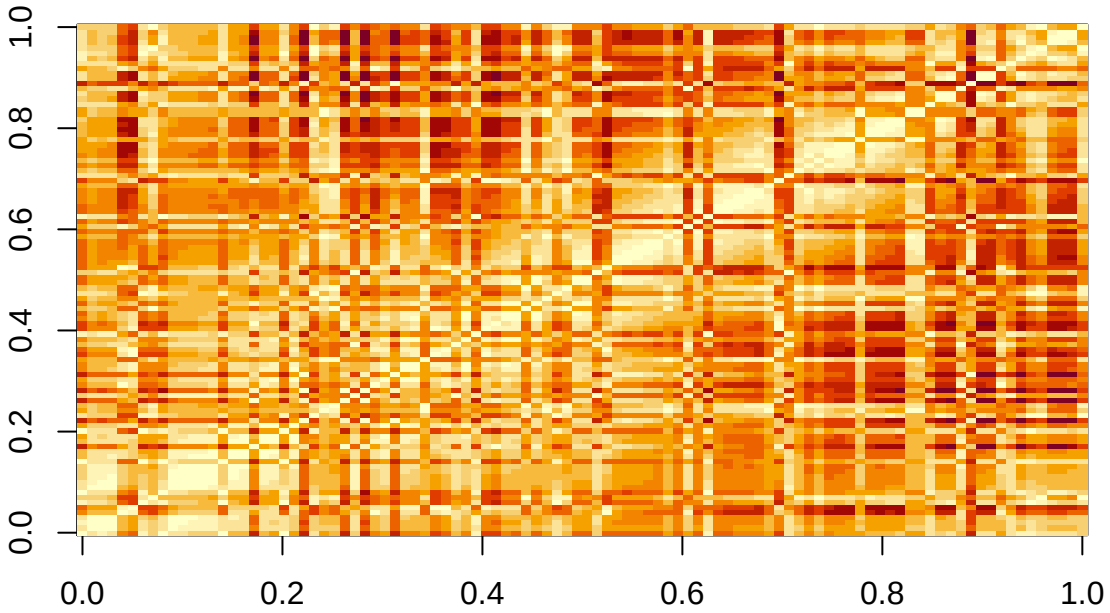
```
head(myData)
```

```
##              X1          X2
## [1,]  2.2439591 -0.87024955
## [2,]  1.1174087  0.06873094
## [3,]  1.4903286  0.15396490
## [4,]  1.3831515  0.35852325
## [5,]  0.6311686  3.26114162
## [6,] -0.6414788  3.38191052
```

```
library(mlbench)

set.seed(111)
obj <- mlbench.spirals(100,1,0.025)
my.data <-  data.frame(4 * obj$x)
names(my.data)<-c("X1","X2")
par(mfrow=c(1,2))
plot(my.data,col=c('orange','blue')[obj$classes],main="Original Classes")
my.data<-as.matrix(my.data)
plot(my.data,col=c('orange','blue')[kmeans(my.data,2)$cluster],main="Kmeans")
```

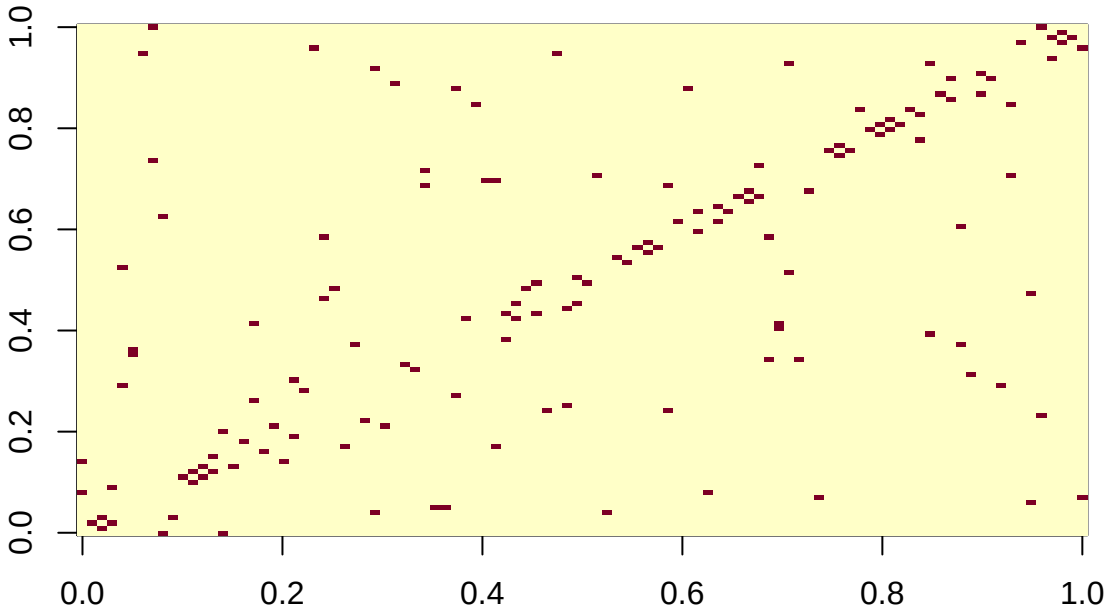



```
Distance.mat <- as.matrix(dist(my.data))
image(Distance.mat)
```




```
diag(Distance.mat) <- max(Distance.mat)
A <- apply(Distance.mat,1,function(x){
  nearest.neig <- which.min(x)
  x[nearest.neig] <- 1
  x[-nearest.neig] <- 0
  return(x)
```

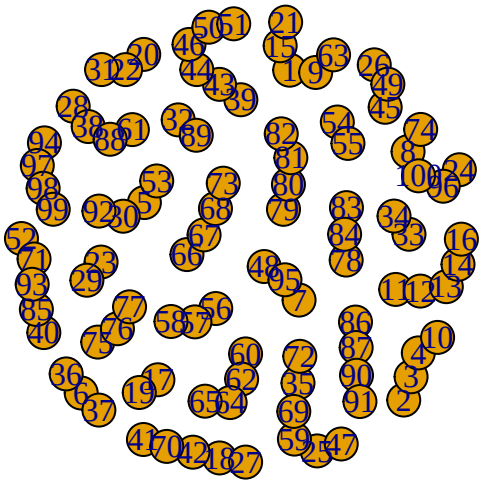
```
})
A <- (A + t(A))>0
image(A)
```



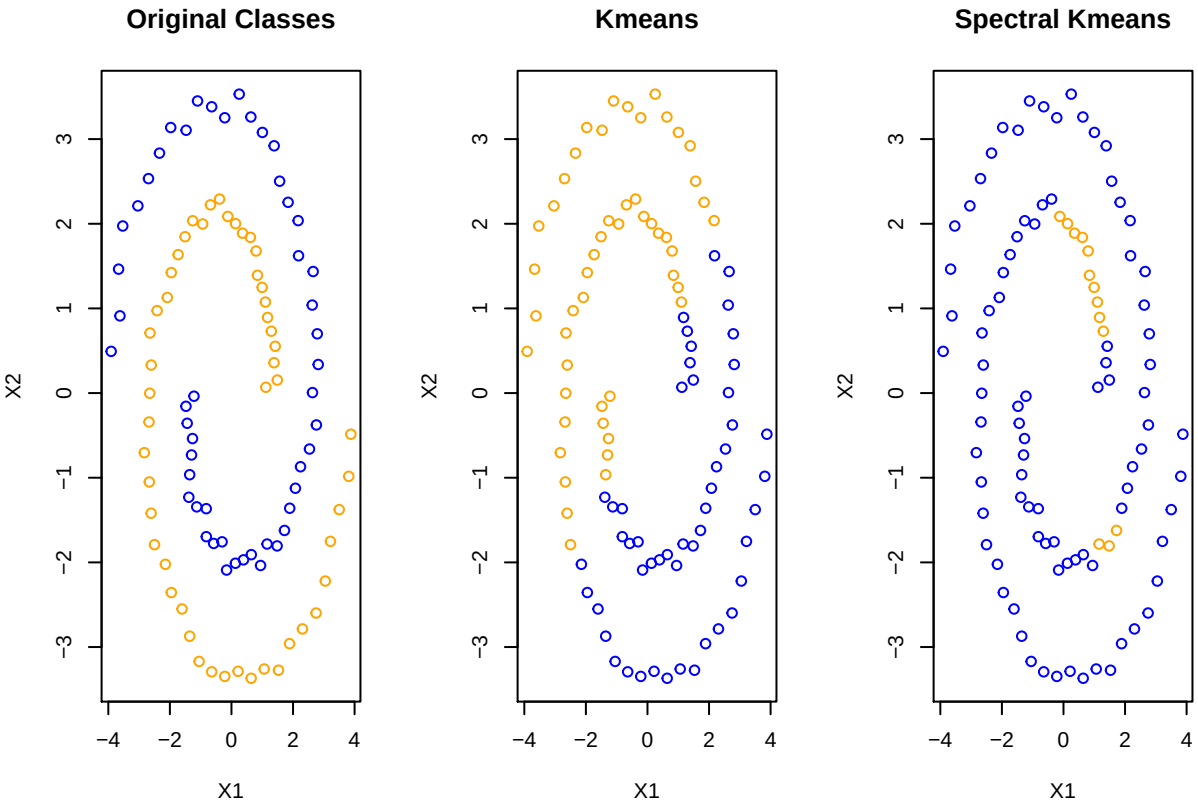

```
library(igraph)
```

```
##
## Attaching package: 'igraph'

## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union
```

```
plot(graph_from_adjacency_matrix(A, mode = "undirected"))
```



```
diag(A)<-0
D<-diag(colSums(A))
L<-D-A
color.kmeans<-kmeans(eigen(L)$vectors[,97:100],2,nstart = 30)$cluster
par(mfrow=c(1,3))
plot(my.data,col=c('orange','blue')[obj$classes],main="Original Classes")
plot(my.data,col=c('orange','blue')[kmeans(my.data,2)$cluster],main="Kmeans")
plot(my.data,col=c('orange','blue')[color.kmeans],main="Spectral Kmeans")
```



2. Compute K the matrix of similarities for this dataset using the gaussian kernel

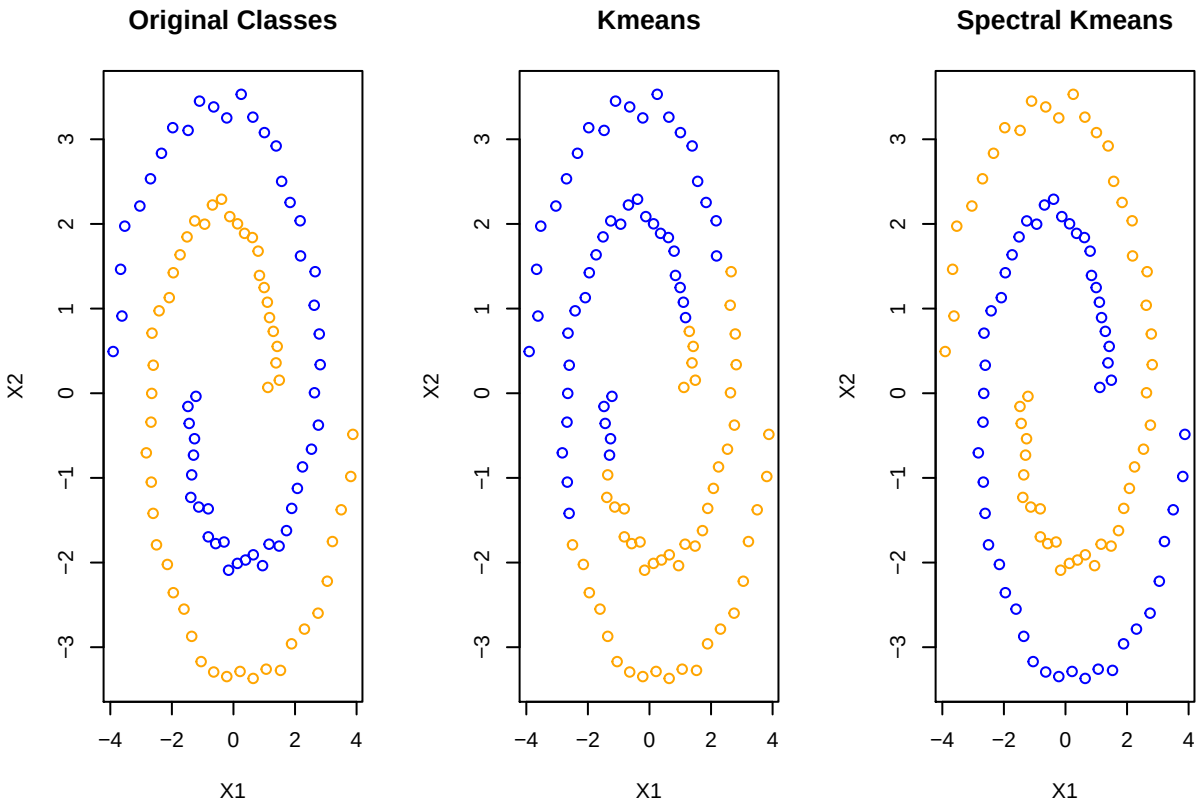
```
sigma2 <- 1
K <- exp(-as.matrix(dist(my.data))^2 / sigma2)
dim(K)
```

```
## [1] 100 100
```

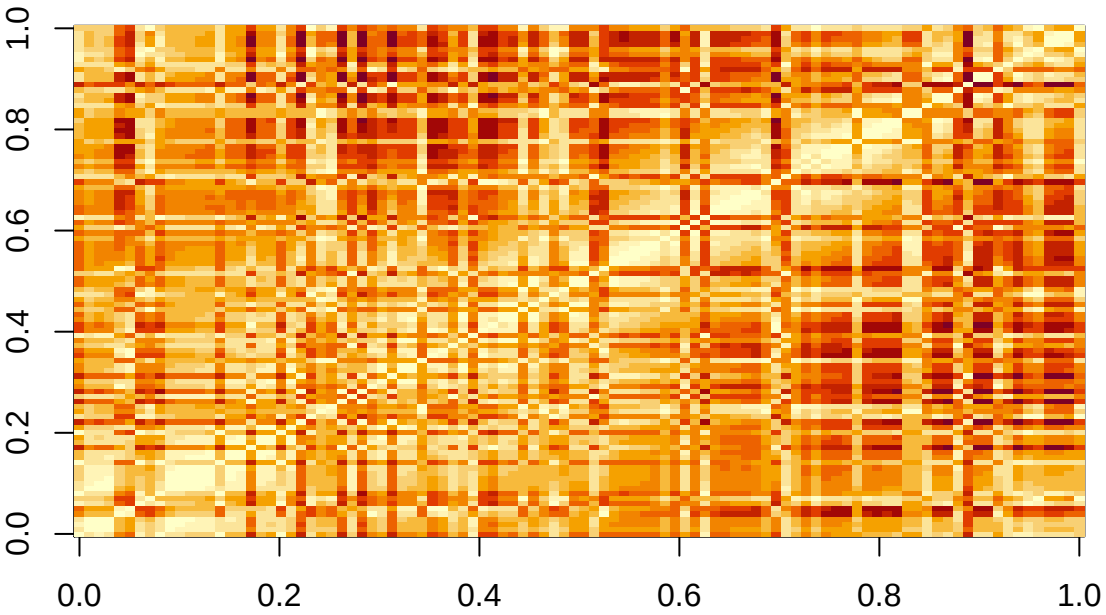
3. The next step consists in computing an affinity matrix A based on K. A must be made of positive values and be symmetric. This is usually done by applying a k-nearest nighboor filter to build a representation of a graph connecting just the closest dataset points. However, to be symmetric, if A_{ij} is selected as a nearest neighboor, so will A_{ji}:

```
A <- K>0.5
diag(A) <- 0
D <- diag(colSums(A))
L <- D-A
```

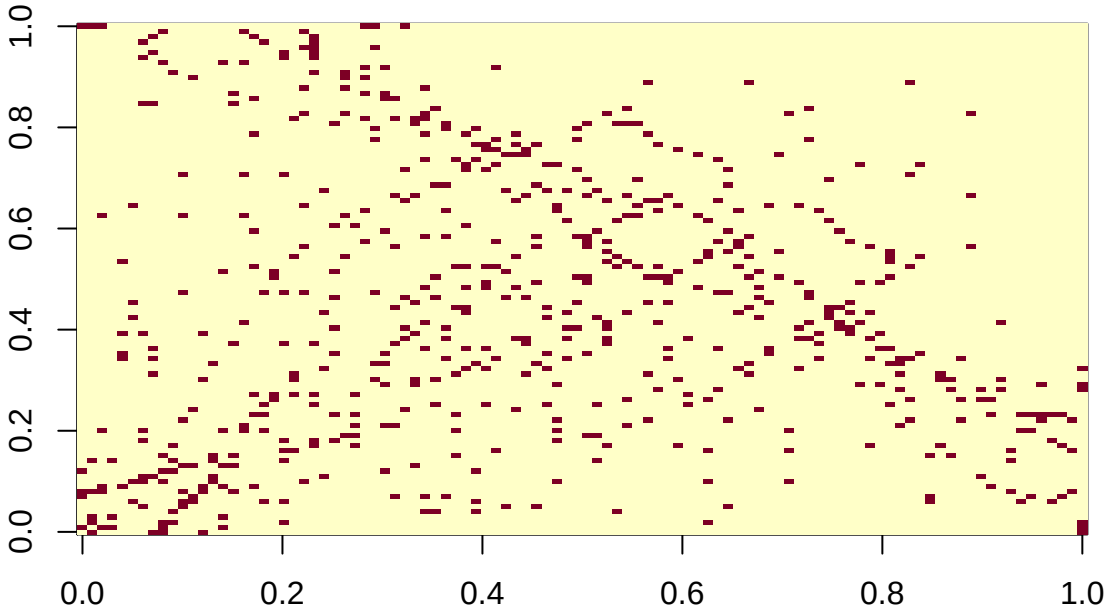
```
color.kmeans <- kmeans(eigen(L)$vectors[,97:100],2,nstart = 30)$cluster
par(mfrow=c(1,3))
plot(my.data,col=c('orange','blue')[obj$classes],main="Original Classes")
plot(my.data,col=c('orange','blue')[kmeans(my.data,2)$cluster],main="Kmeans")
plot(my.data,col=c('orange','blue')[color.kmeans],main="Spectral Kmeans")
```



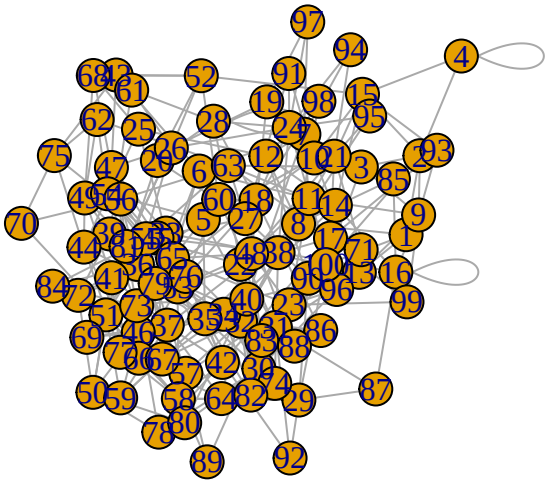

```
Distance.mat <- as.matrix(dist(my.data))
image(Distance.mat)
```



```
diag(Distance.mat) <- max(Distance.mat)
A <- apply(Distance.mat,1,function(x){
  nearest.neig <- rank(x)[1:3]
  x[nearest.neig] <- 1
  x[-nearest.neig] <- 0
  return(x)
})
A <- (A + t(A))>0
image(A)
```

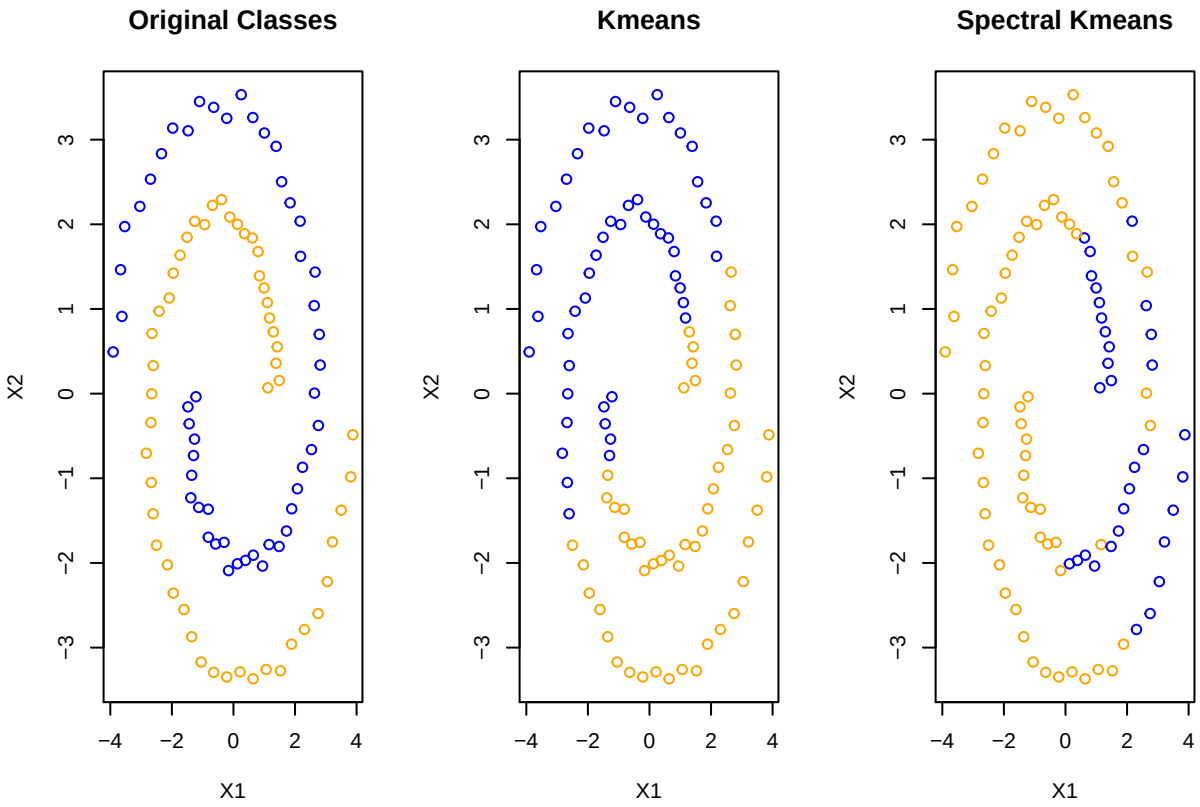



```
library(igraph)
plot(graph_from_adjacency_matrix(A, mode = "undirected"))
```



```
diag(A)<-0
D<-diag(colSums(A))
L<-D-A
color.kmeans<-kmeans(eigen(L)$vectors[,98:100],2,nstart = 30)$cluster
```

```
par(mfrow=c(1,3))
plot(my.data,col=c('orange','blue')[obj$classes],main="Original Classes")
plot(my.data,col=c('orange','blue')[kmeans(my.data,2)$cluster],main="Kmeans")
plot(my.data,col=c('orange','blue')[color.kmeans],main="Spectral Kmeans")
```

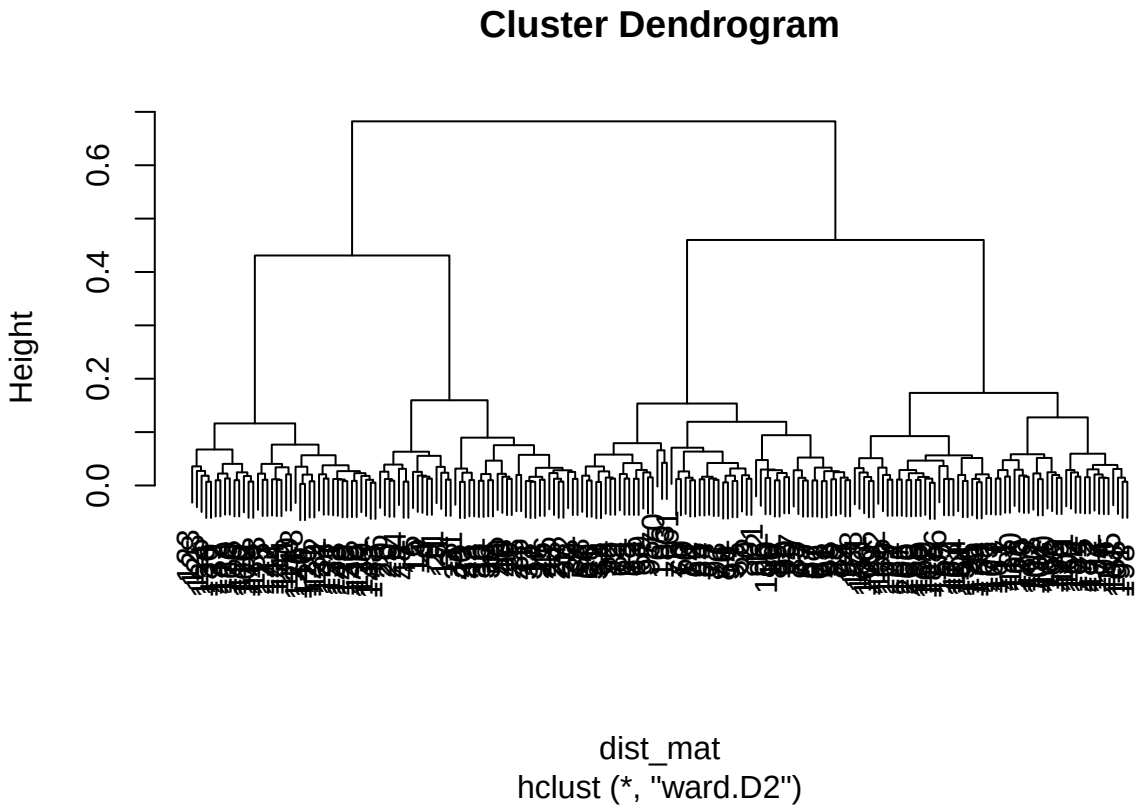


Examples

Normalized crabs

Hierarchical agglomerative clustering with R base

```
dist_mat<-dist(crabsquant2)
hclust_ward.D2 <- hclust(dist_mat, method = 'ward.D2')
plot(hclust_ward.D2)
```



Cutting the tree

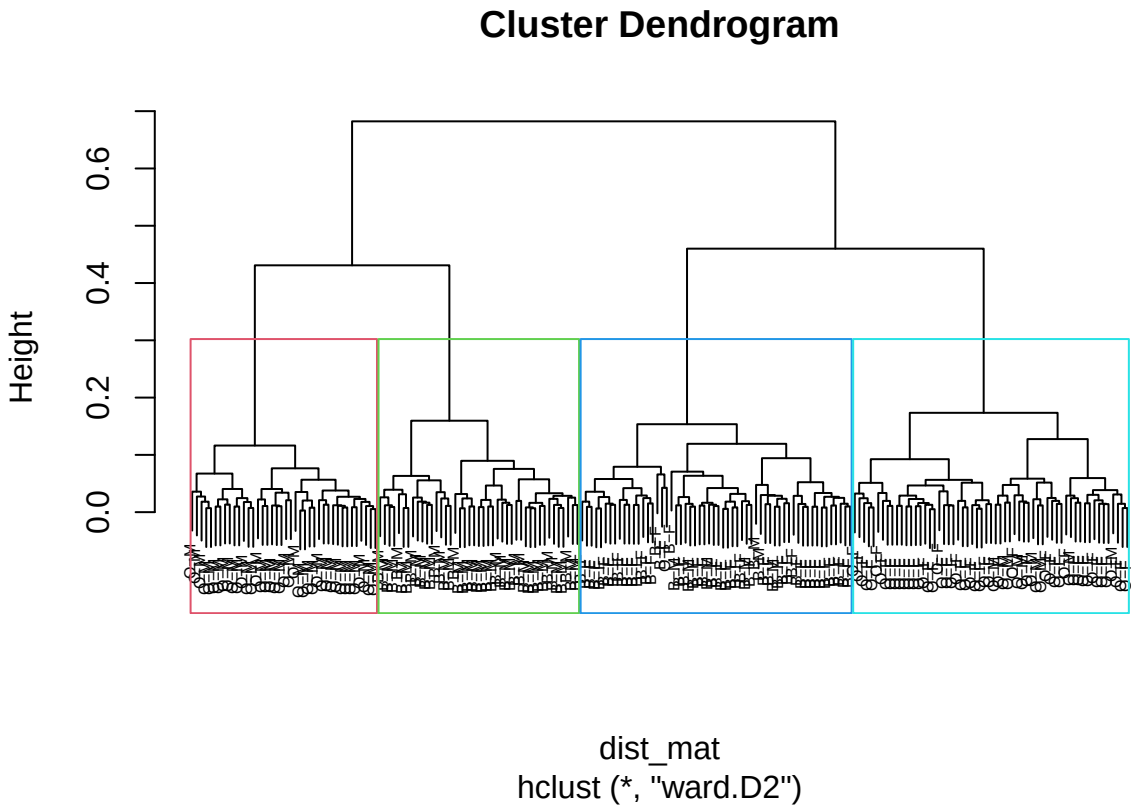
to get a partition

```
cut_ward.D2 <- cutree(hclust_ward.D2,k = 4)
table(true.classes,cut_ward.D2)
```

```
##             cut_ward.D2
## true.classes  1  2  3  4
##          B-F 50  0  0  0
##          B-M  7 43  0  0
##          O-F  1  0 49  0
##          O-M  0  0 10 40
```

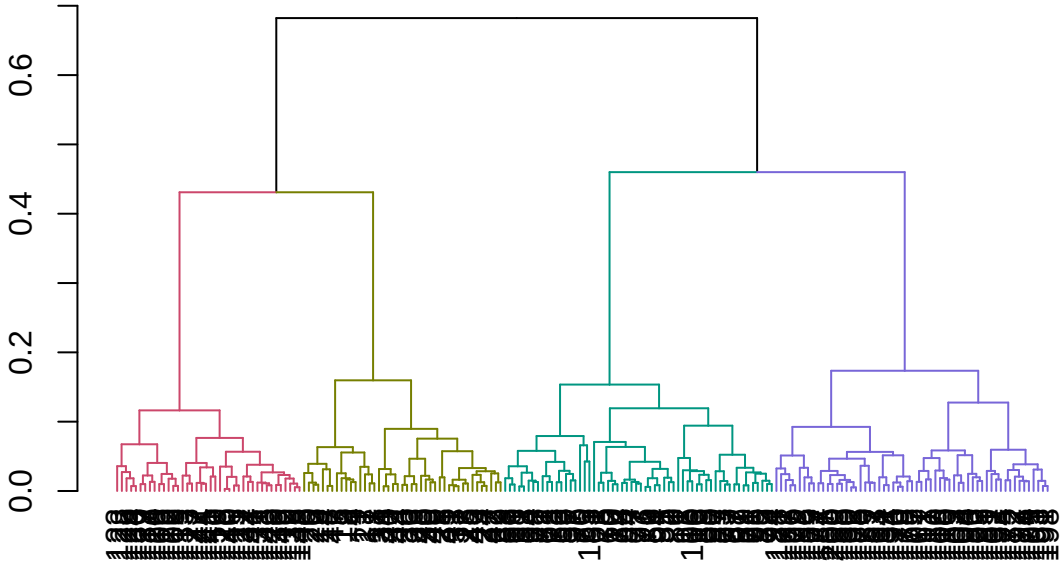
Display partition in dendrogram

```
plot(hclust_ward.D2,labels = true.classes,cex=.5)
rect.hclust(hclust_ward.D2 , k = 4, border = 2:6)
```



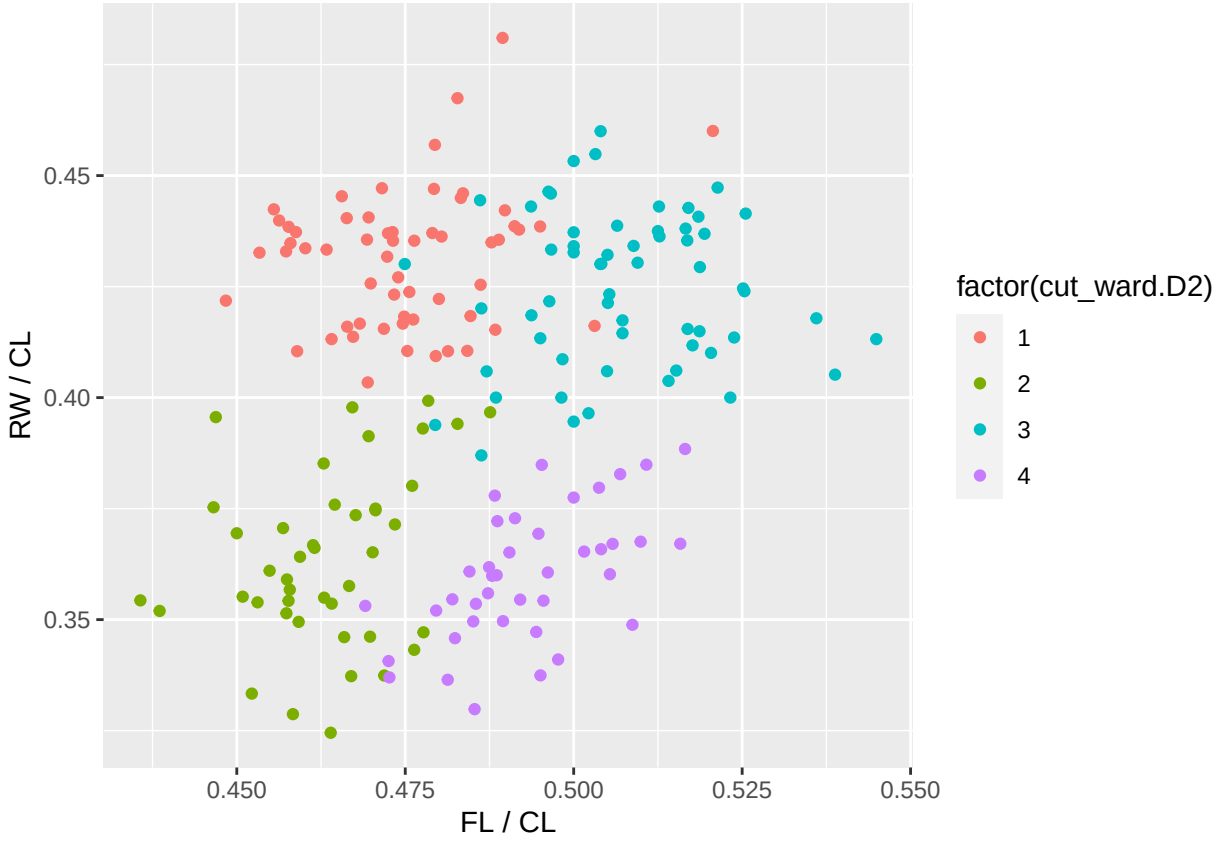
Using package "dendextend"

```
suppressPackageStartupMessages(library(dendextend))
ward.D2_dend_obj <- as.dendrogram(hclust_ward.D2)
ward.D2_col_dend <- color_branches(ward.D2_dend_obj, k = 4)
plot(ward.D2_col_dend)
```



Biplot of the hierachical clutering for 4 clusters

```
suppressPackageStartupMessages(library(ggplot2))
ggplot(crabsquant2, aes(x=`FL / CL`, y = `RW / CL`, color = factor(cut_ward.D2))) + geom_point()
```



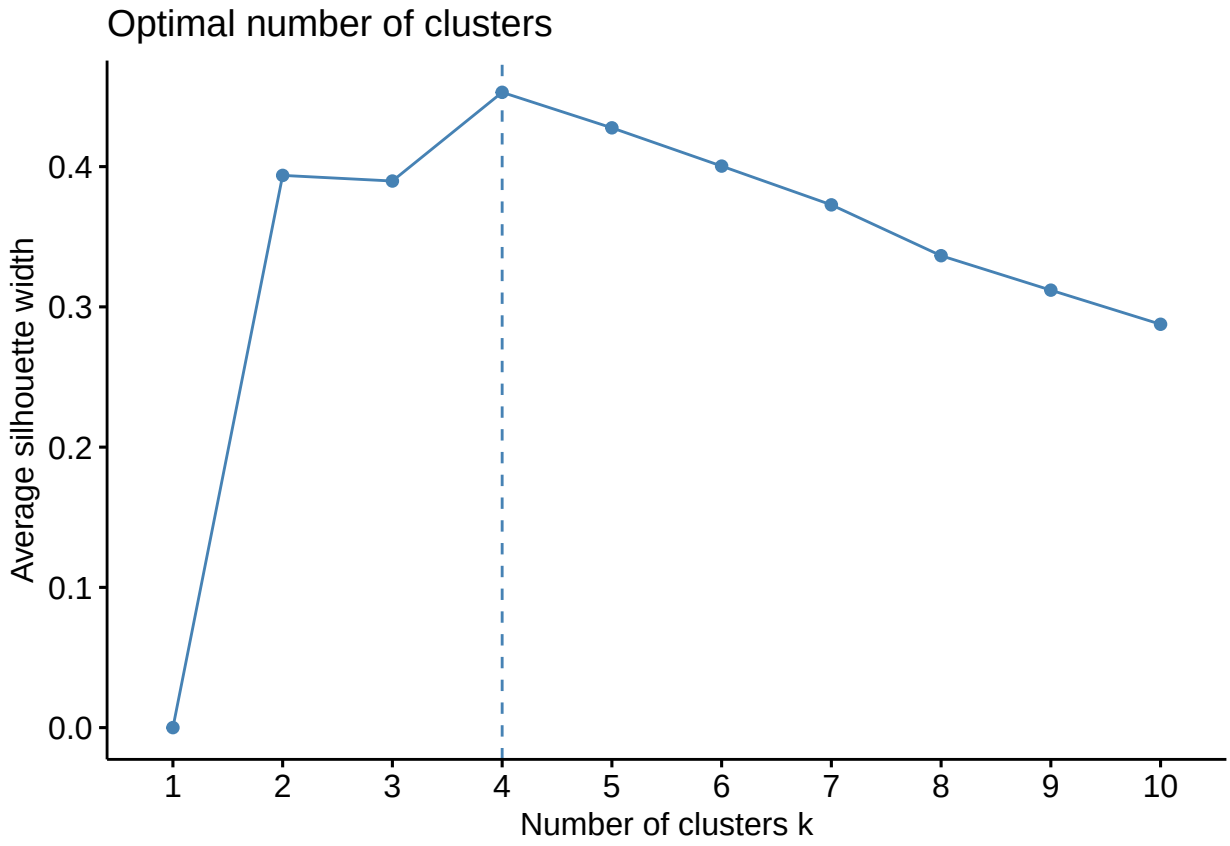
Dertermining and Visualizing the Optimal Number of Clusters

Silhouette

```
library(factoextra)
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

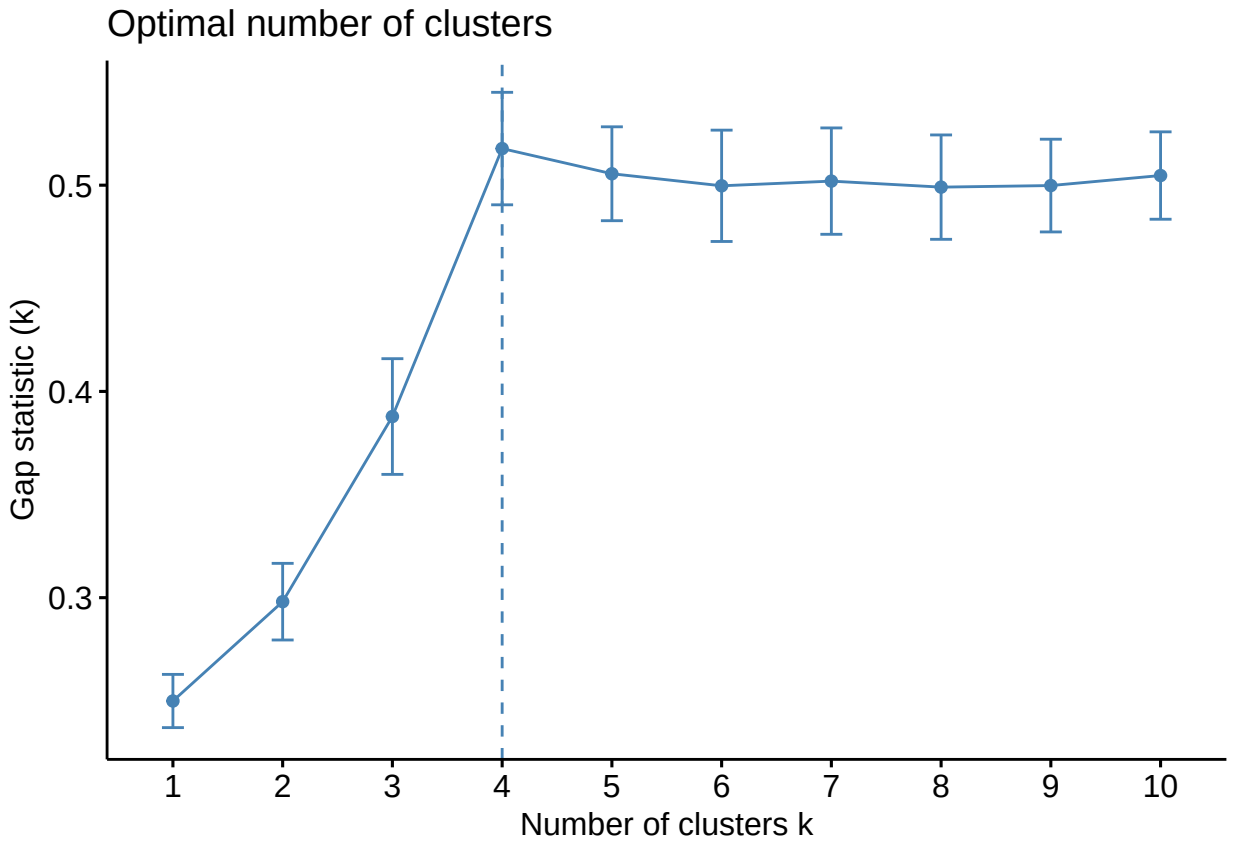
```
fviz_nbclust(as.matrix(dist_mat), FUN = hcut, method = "silhouette")
```



Dertermining and Visualizing the Optimal Number of Clusters

Gap Statistics

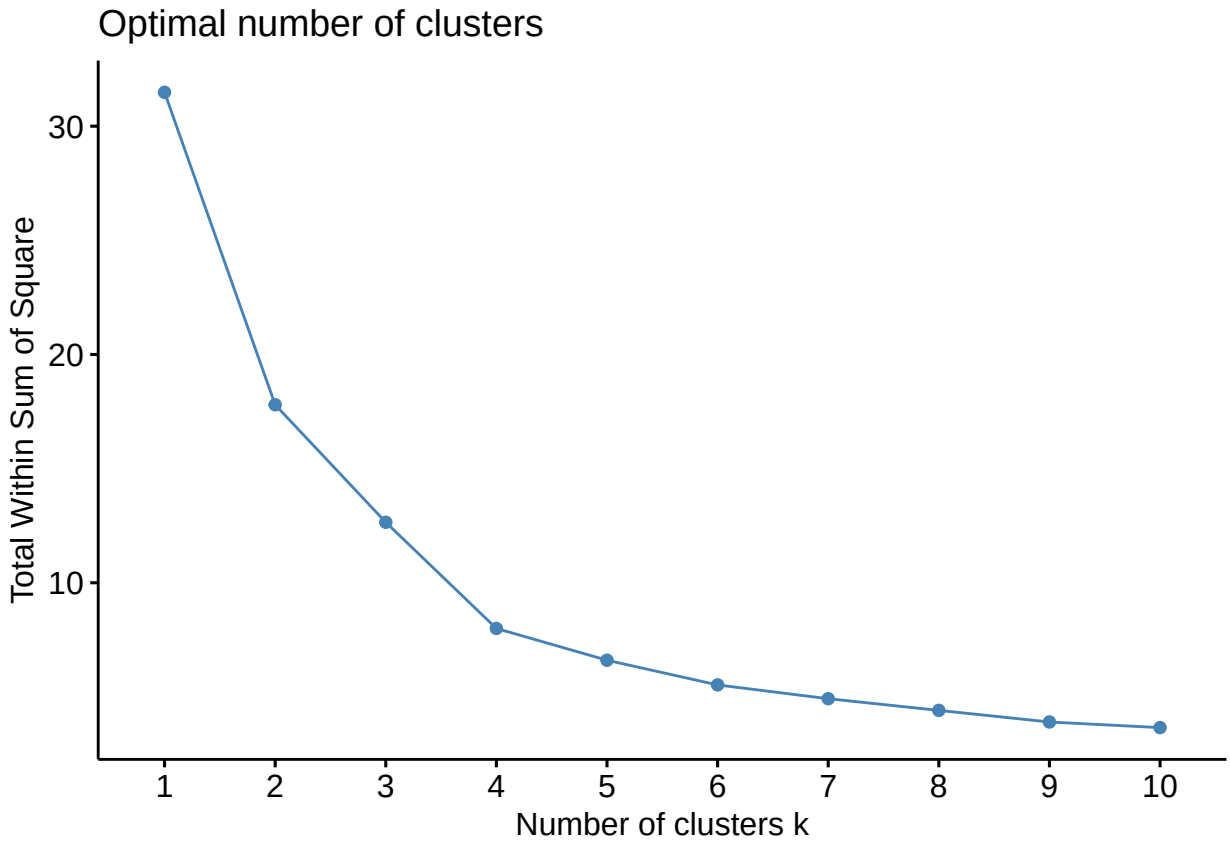
```
gap_stat <- cluster::clusGap(crabsquant2, FUN = hcut, K.max = 10, B = 10)
fviz_gap_stat(gap_stat)
```



Dertermining and Visualizing the Optimal Number of Clusters

wss

```
library(factoextra)
fviz_nbclust(as.matrix(dist_mat), FUN = hcut, method = "wss")
```



Hierarchical agglomerative clustering with R package cluster

agnes

> Agglomerative Nesting Description: Computes agglomerative hierarchical clustering of the dataset. Usage: agnes(x, diss = inherits(x, "dist"), metric = "euclidean", stand = FALSE, method = "average", keep.diss = n < 100, keep.data = !diss)

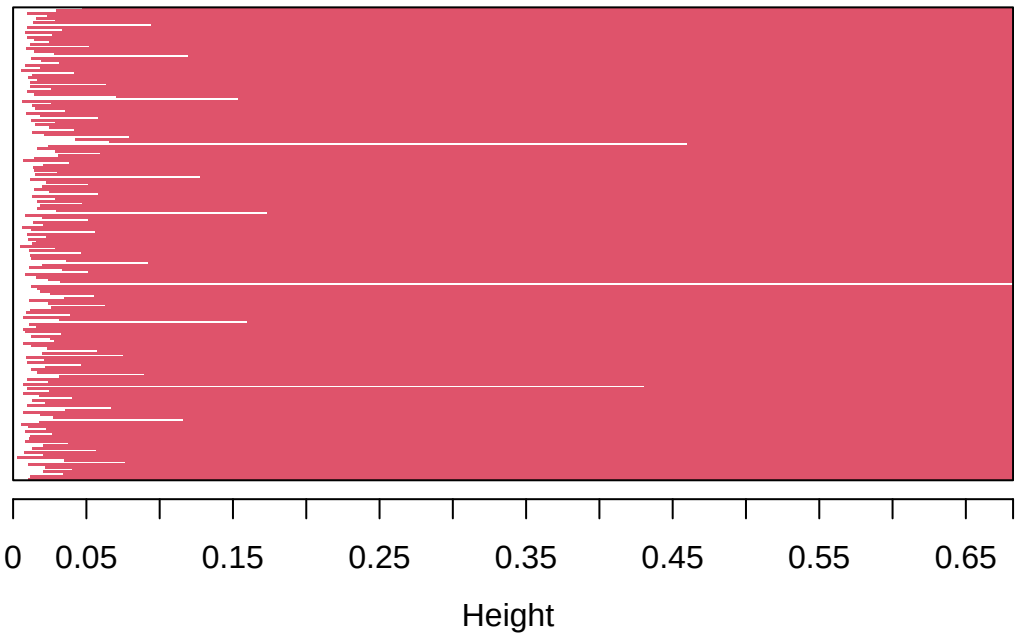
Agglomerative clustering with Ward

```
library(cluster)
res<-agnes(crabsquant2,method="ward")
```

Banner plot

```
plot(res,which.plots=1)
```

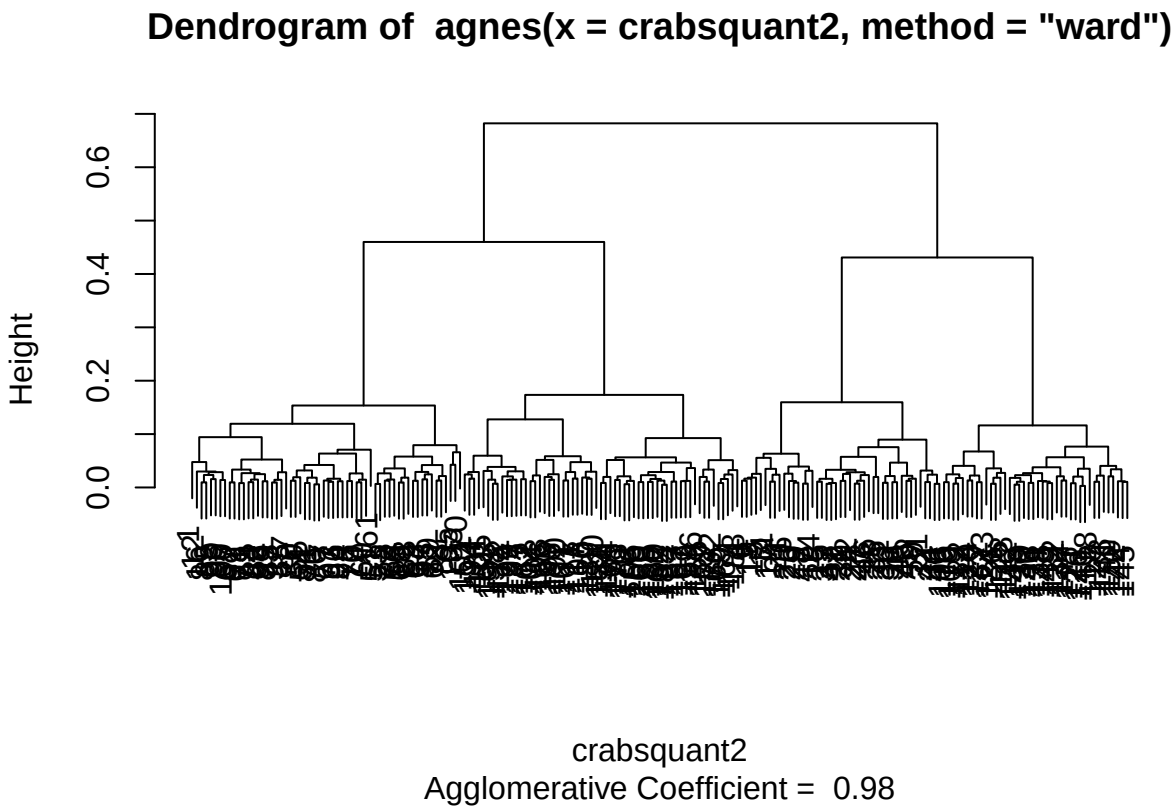
Banner of agnes(x = crabsquant2, method = "ward")



Agglomerative Coefficient = 0.98

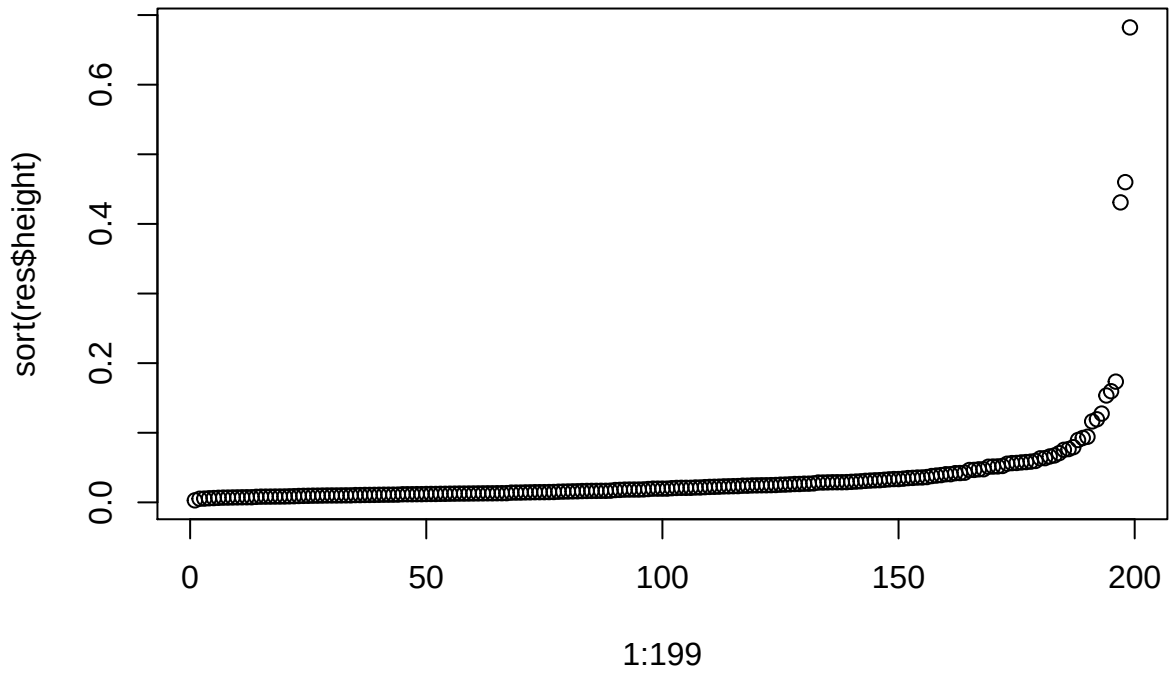
Dendogram

```
plot(res,which.plots=2)
```



Levels of fusion

```
plot(1:199,sort(res$height))
```



Divisive Hiearchical Clustering with R

diana

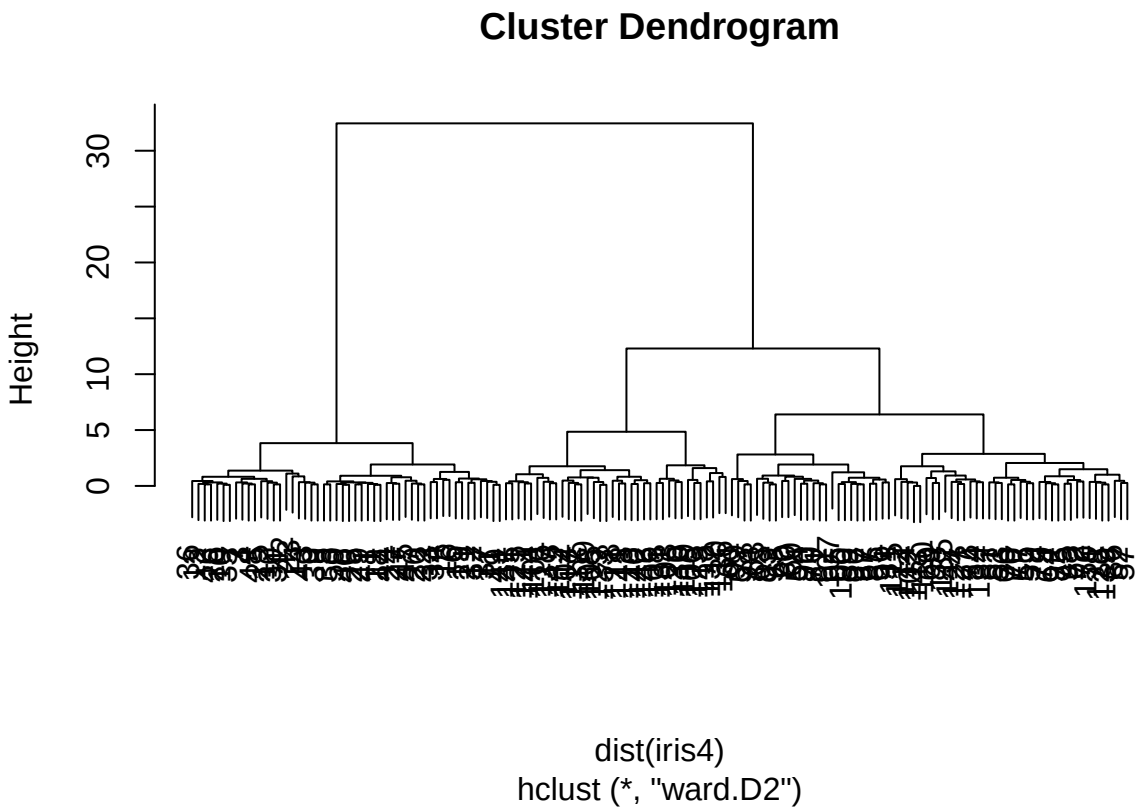
> DIvisive ANAlysis Clustering Description: Computes a divisive hierarchical clustering of the dataset returning an object of class 'diana'. Usage:
>
> diana(x, diss = inherits(x, "dist"), metric = "euclidean", stand = >FALSE, keep.diss = n < 100, keep.data = !diss)

```
# Another questions from the lesson

Load the iris data set and try hierarchical clustering with all available fusion strategies. Compare th
```

```r
library(MASS)
data(iris)
dim(iris)
```

```
## [1] 150   5
```

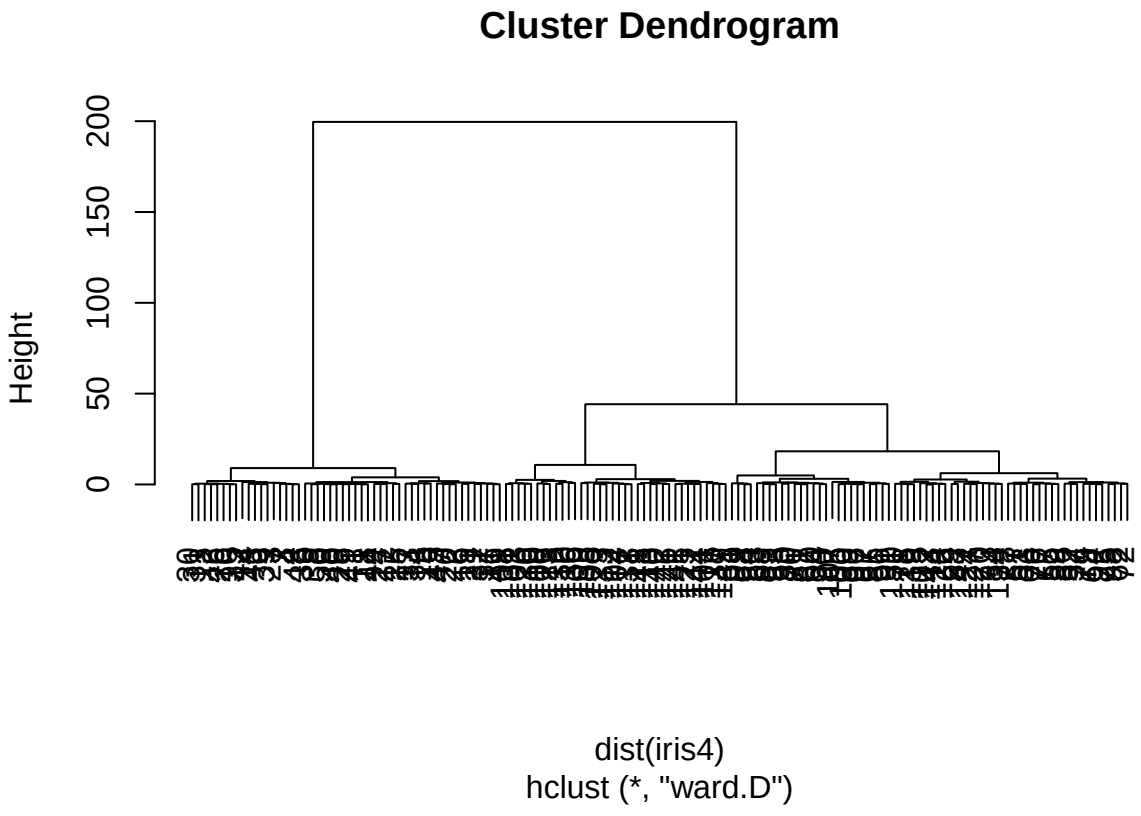
```r
iris4<-iris[,1:4]
hclust_ward.D2 <- hclust(dist(iris4), method = 'ward.D2')
plot(hclust_ward.D2)
```



```
true.classes<-iris$Species
cut_ward.D2 <- cutree(hclust_ward.D2,k = 3)
table(true.classes,cut_ward.D2)
```

```
##             cut_ward.D2
## true.classes  1  2  3
##   setosa     50  0  0
##   versicolor  0 49  1
##   virginica   0 15 35
```

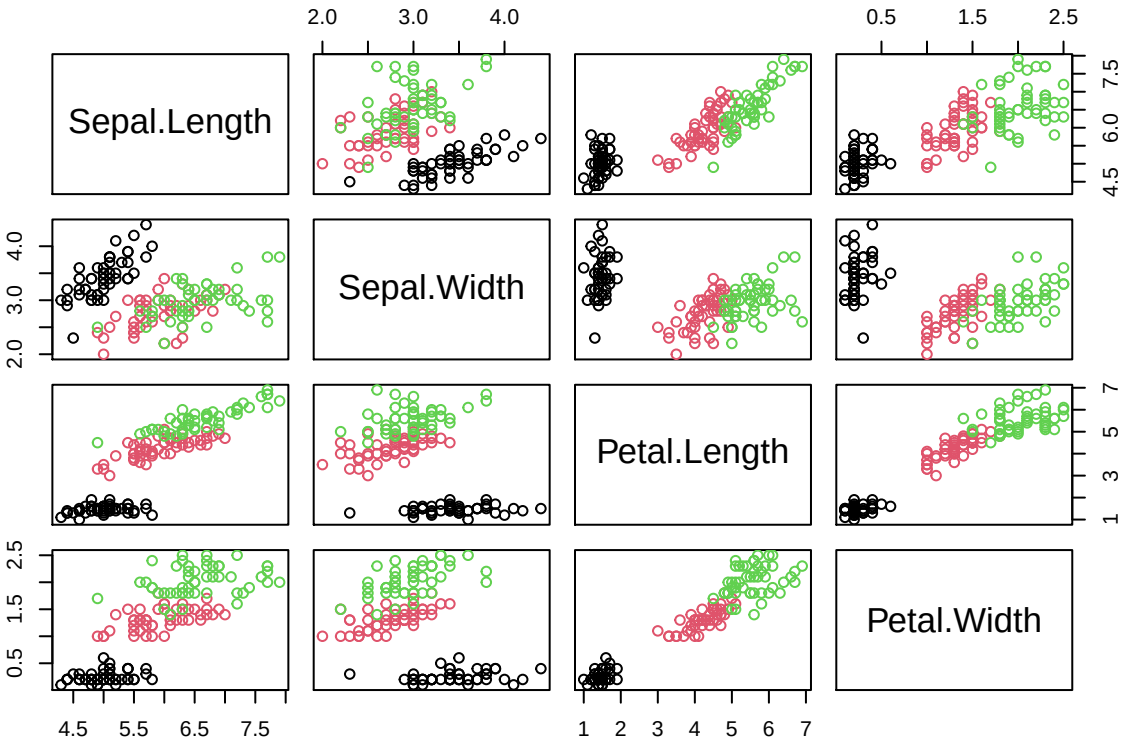
```
iris4<-iris[,1:4]
hclust_ward.D <- hclust(dist(iris4), method = 'ward.D')
plot(hclust_ward.D)
```



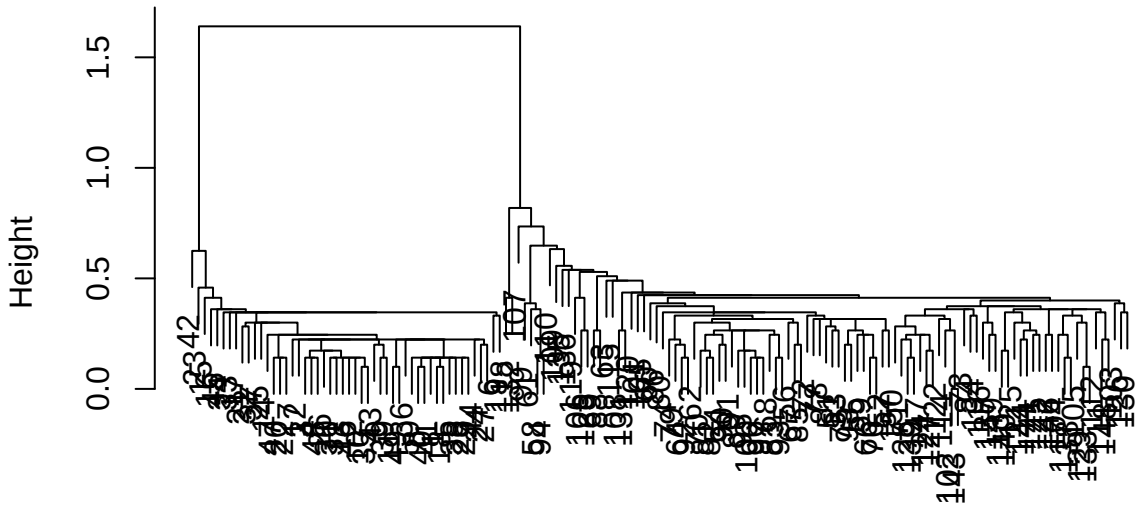

```
true.classes<-iris$Species
cut_ward.D <- cutree(hclust_ward.D,k = 3)
table(true.classes,cut_ward.D)
```

```
##             cut_ward.D
## true.classes  1  2  3
##   setosa     50  0  0
##   versicolor  0 50  0
##   virginica   0 14 36
```

```
pairs(iris4, col=iris$Species)
```



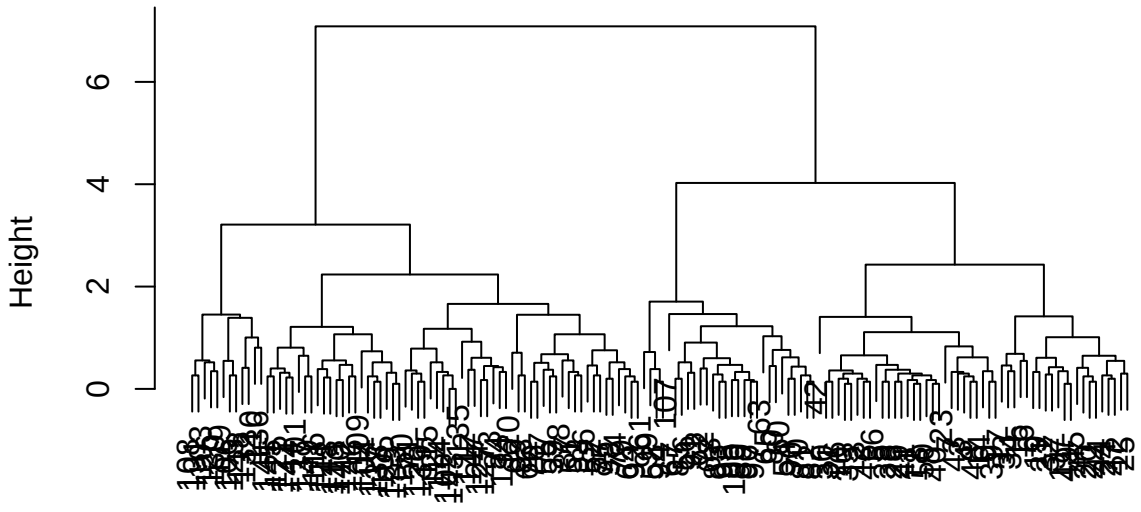

```
hclust_single <- hclust(dist(iris4), method = 'single') # single link => min
plot(hclust_single)
```




```
true.classes<-iris$Species
cut_single <- cutree(hclust_single,k = 3)
table(true.classes,cut_single) # almost only 2 classes
```

```
##             cut_single
## true.classes  1  2  3
##   setosa     50  0  0
##   versicolor  0 50  0
##   virginica   0 48  2
```

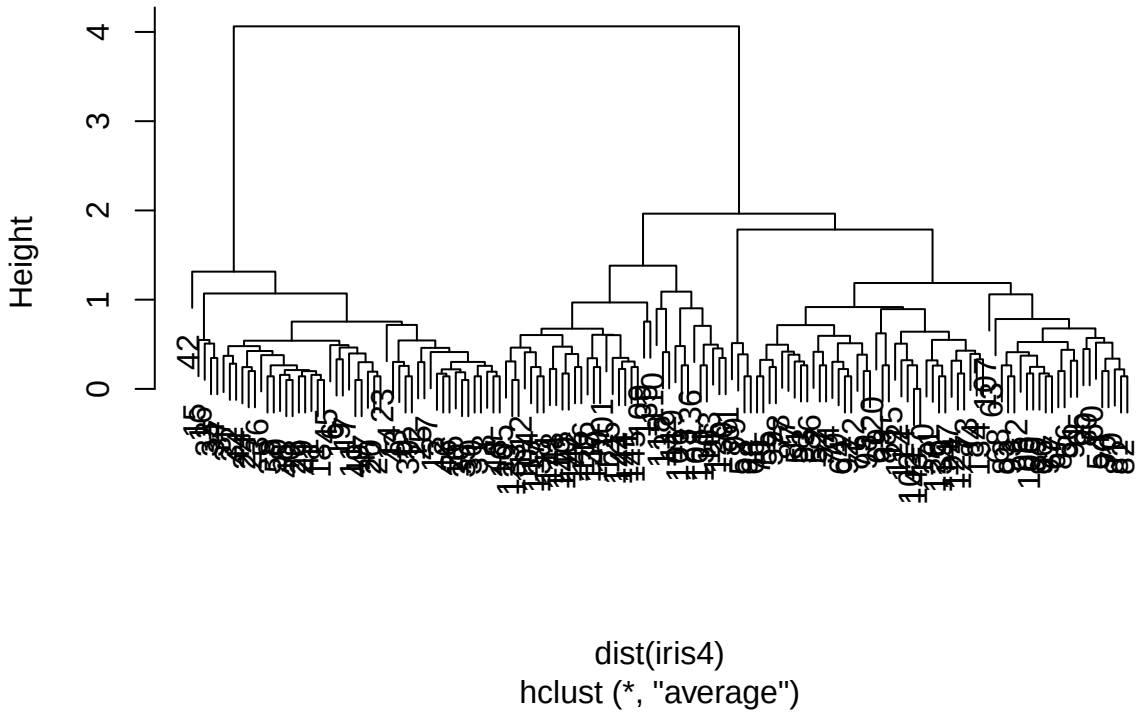
```
hclust_complete <- hclust(dist(iris4), method = 'complete')
plot(hclust_complete)
```




```
true.classes<-iris$Species
cut_complete <- cutree(hclust_complete,k = 3)
table(true.classes,cut_complete)
```

```
##              cut_complete
## true.classes  1  2  3
##    setosa     50  0  0
##    versicolor  0 23 27
##    virginica   0 49  1
```

```
hclust_average <- hclust(dist(iris4), method = 'average')
plot(hclust_average)
```



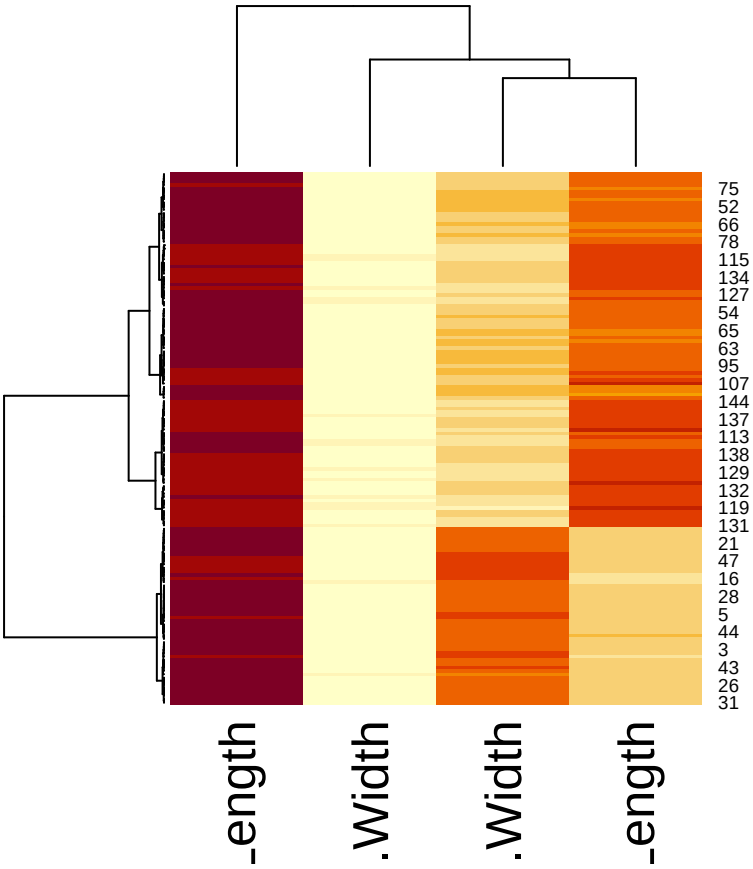

```
true.classes<-iris$Species
cut_average <- cutree(hclust_average,k = 3)
table(true.classes,cut_average)
```

```
##             cut_average
## true.classes  1  2  3
##   setosa     50  0  0
##   versicolor  0 50  0
##   virginica   0 14 36
```

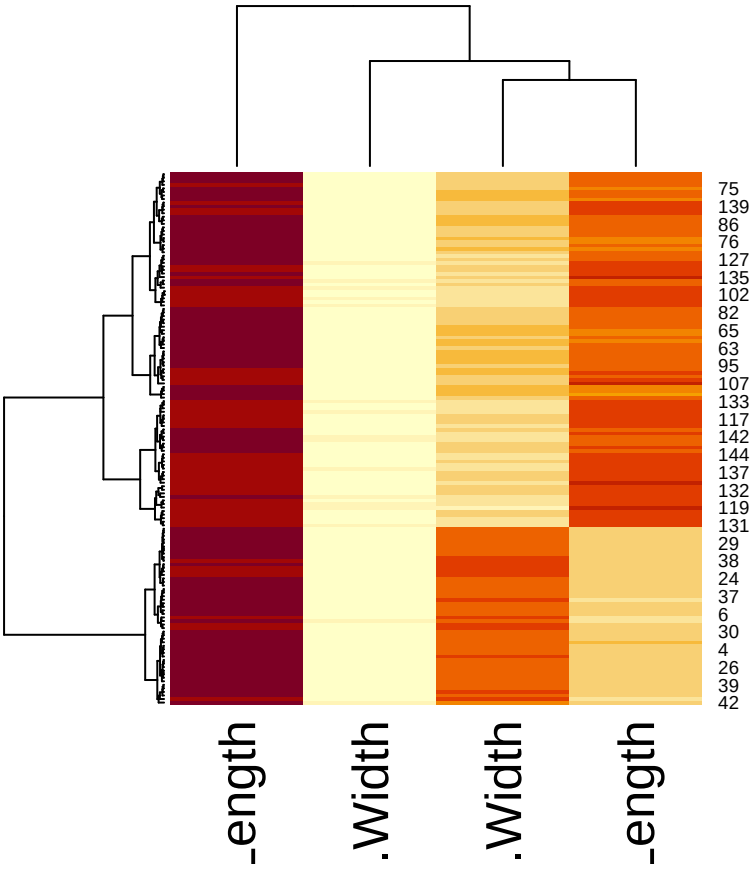
Heatmap

Try heatmap to cluster both rows and columns and then display the results

```
heatmap(as.matrix(iris4), hclustfun = function(x) hclust(x, method="ward.D"))
```

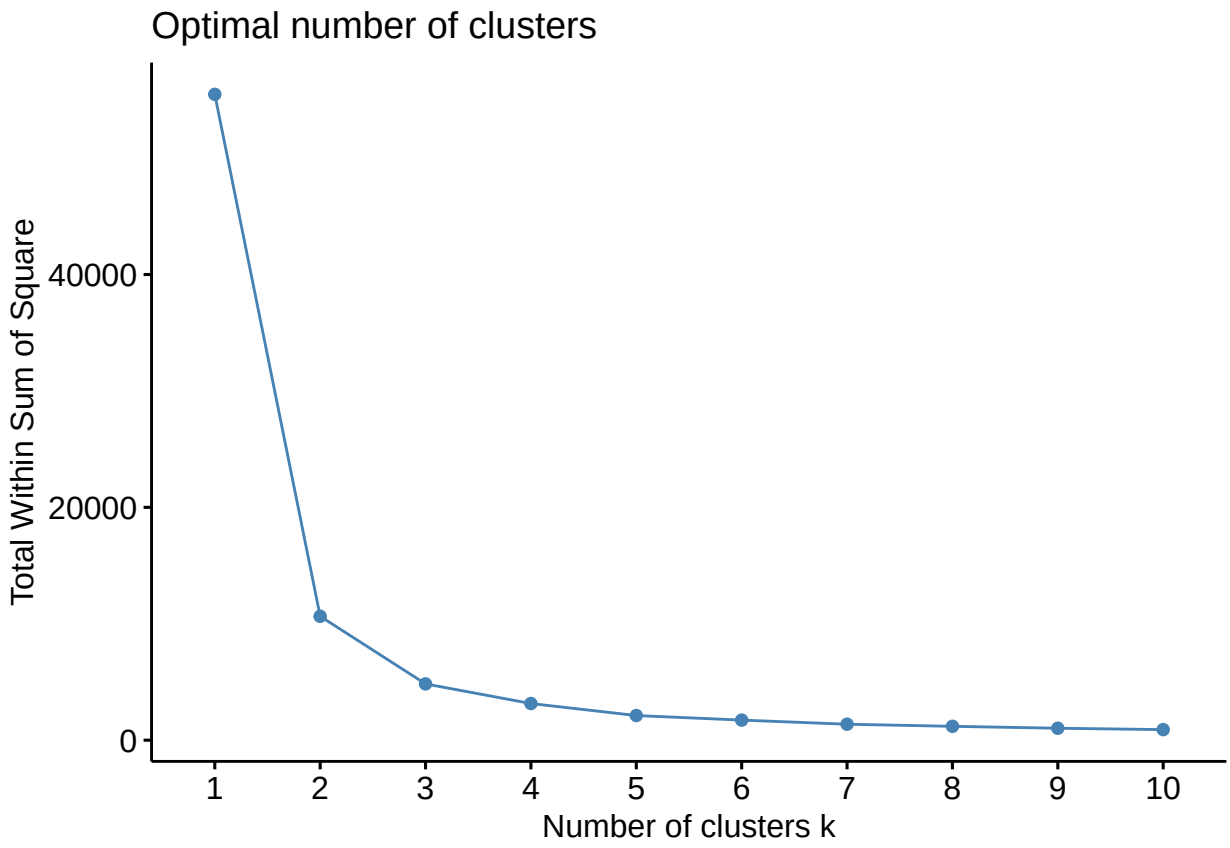



```
heatmap(as.matrix(iris4), hclustfun = function(x) hclust(x, method="ward.D2"))
```

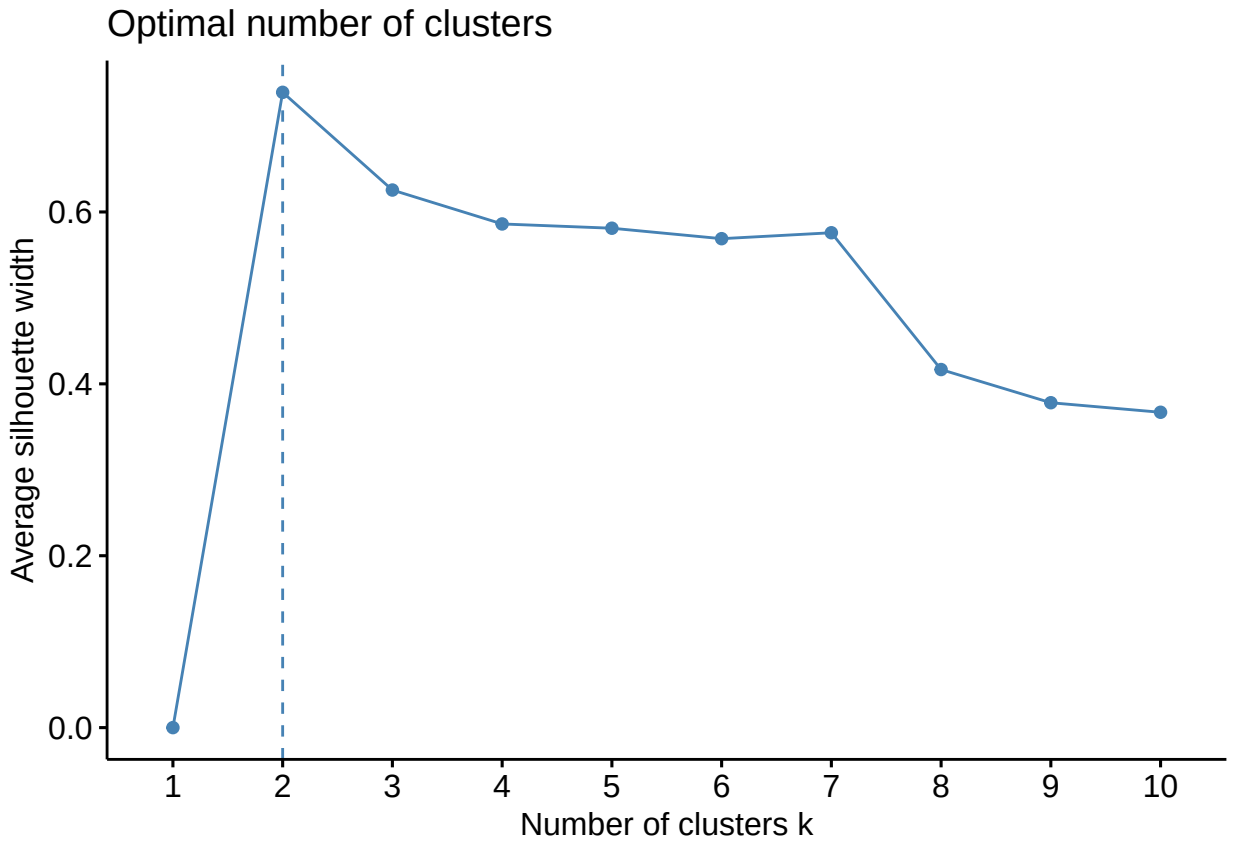


3. Choose the number of classes using Ward.D and silhouette algorithm.

```
library(factoextra)
library(ggplot2)
fviz_nbclust(as.matrix(dist(iris4)), FUNcluster = hcut, method = "wss", hc_method = "ward.D")
```



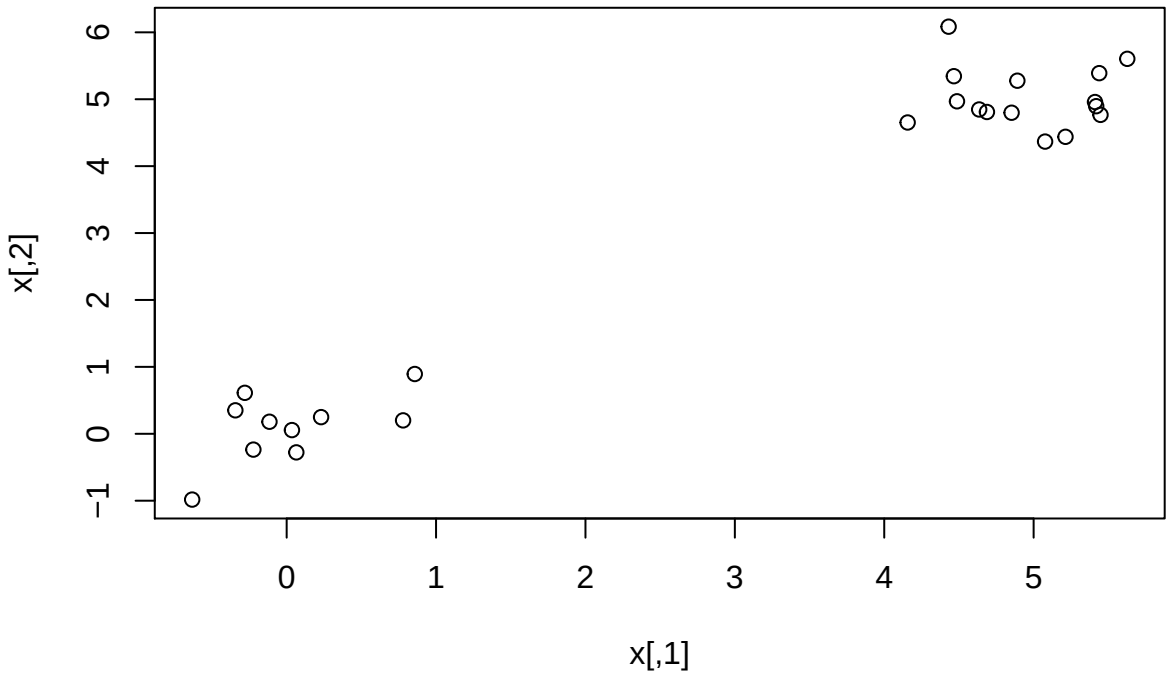

```
fviz_nbclust(as.matrix(dist(iris4)), FUNcluster = hcut, method = "silhouette", hc_method = "ward.D")
```



Partition Around Medoids

The variant of k-means when data is available as dissimilarity matrix.

```
library(cluster)
x <- rbind(cbind(rnorm(10,0,0.5), rnorm(10,0,0.5)),
           cbind(rnorm(15,5,0.5), rnorm(15,5,0.5)))
plot(x)
```




```
pamx <- pam(x,2)
pamx
```

```
## Medoids:
##      ID
## [1,]  4 0.0352542 0.05534136
## [2,] 22 4.8524643 4.79855758
## Clustering vector:
##  [1] 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## Objective function:
##     build      swap
## 0.5823017 0.5673078
##
## Available components:
##  [1] "medoids"    "id.med"     "clustering" "objective"  "isolation"
##  [6] "clusinfo"   "silinfo"    "diss"       "call"       "data"
```

```
summary(pamx)
```

```
## Medoids:
##      ID
## [1,]  4 0.0352542 0.05534136
## [2,] 22 4.8524643 4.79855758
## Clustering vector:
##  [1] 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## Objective function:
##     build      swap
## 0.5823017 0.5673078
##
## Numerical information per cluster:
##      size max_diss   av_diss diameter separation
```

```
## [1,]   10 1.234800 0.5481373 2.396359    5.00157
## [2,]   15 1.353248 0.5800881 1.834583    5.00157
##
## Isolated clusters:
##  L-clusters: character(0)
##  L*-clusters: [1] 1 2
##
## Silhouette plot information:
##    cluster neighbor sil_width
## 4        1        2 0.9129369
## 2        1        2 0.9114441
## 7        1        2 0.9006804
## 10       1        2 0.8986915
## 5        1        2 0.8977615
## 9        1        2 0.8933949
## 1        1        2 0.8743677
## 3        1        2 0.8439481
## 8        1        2 0.8226917
## 6        1        2 0.7740986
## 22       2        1 0.9076739
## 12       2        1 0.9041405
## 15       2        1 0.9021105
## 25       2        1 0.9009921
## 14       2        1 0.9002473
## 17       2        1 0.8997290
## 23       2        1 0.8924435
## 13       2        1 0.8892038
## 24       2        1 0.8873568
## 21       2        1 0.8777867
## 11       2        1 0.8774858
## 18       2        1 0.8714328
## 20       2        1 0.8627011
## 16       2        1 0.8307087
## 19       2        1 0.8170298
## Average silhouette width per cluster:
## [1] 0.8730015 0.8814028
## Average silhouette width of total data set:
## [1] 0.8780423
##
## 300 dissimilarities, summarized :
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0628  0.7777  3.6990  3.9039  6.9271  9.0870
## Metric :  euclidean
## Number of objects : 25
##
## Available components:
##  [1] "medoids"    "id.med"     "clustering" "objective"  "isolation"
##  [6] "clusinfo"   "silinfo"    "diss"       "call"       "data"
```

SBM Model

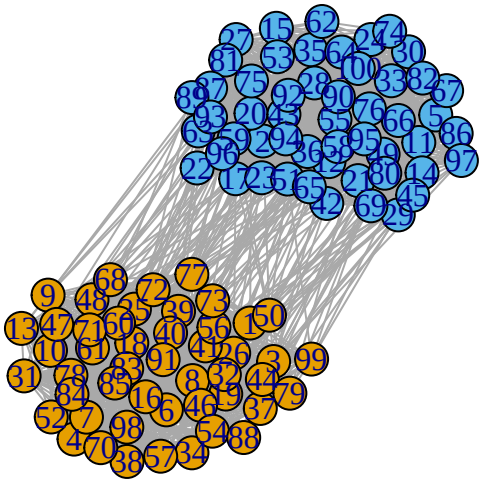
Algorithm of simulation:

Write an R function to simulate an affiliation network (special case of stochastic block model)

```
class.ind<-function (cl)
{
    n <- length(cl)
    cl <- as.factor(cl)
    x <- matrix(0, n, length(levels(cl)))
    x[(1:n) + n * (unclass(cl) - 1)] <- 1
    dimnames(x) <- list(names(cl), levels(cl))
    x
}

class.ind(c(1,1,1,2,2))
```

```
##      1 2
## [1,] 1 0
## [2,] 1 0
## [3,] 1 0
## [4,] 0 1
## [5,] 0 1
```

```
graph.affiliation<-function(n=100,Pi=c(1/2,1/2),alpha=0.7,beta=0.05) {
      # INPUT  n: number of vertex
      #           Pi : vecteur of class proportion
      #           alpha: proba of edge given  same class
      #           beta: proba of edge given two different classes
      # OUTPUT x: adjacency matrix
      #        cluster: class vector
      #

      X<-matrix(0,n,n); # reserve space for adjacency matrix
      Q<-length(Pi);
      rmultinom(1, size=n, prob = Pi)->nq;
      Z<-class.ind(rep(1:Q,nq));
      Z<-Z[sample(1:n,n),];
      for (i in 1:n)
        for (j in i:n)
            {
            # if i and j in same class
            if (which.max(Z[i,])  == which.max(Z[j,])) p<-alpha else  p<-beta
            if ((rbinom(1,1,p))&(i != j)) {X[i,j]<-1; X[j,i]<-1}
            }
       return(list(X=X,cluster=apply(Z,1,which.max)) )
  }

mygraph<-graph.affiliation(alpha=0.7,beta=0.05)
library(igraph)
plot(graph_from_adjacency_matrix(mygraph$X,mode="undirected"),vertex.color=mygraph$cluster)
```



```
matrix(0,5,7)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7]
## [1,]    0    0    0    0    0    0    0
## [2,]    0    0    0    0    0    0    0
## [3,]    0    0    0    0    0    0    0
## [4,]    0    0    0    0    0    0    0
## [5,]    0    0    0    0    0    0    0
```