Aprendizaje No Supervisado

Eric Bellet

2 de Abril de 2016

Introducción

El proceso de aprendizaje cuando desconocemos la columna clase se denomina en la literatura **aprendizaje no supervisado**. En algunos de estos casos, podemos aplicar algoritmos que nos permitan encontrar, de existir, estructuras que denominamos clústers para que, una vez encontrados, podamos analizarlos uno a uno hasta conseguir características de relevancia que nos permitan solucionar el problema planteado originalmente.

Este subconjunto de tareas donde podemos encontrar estructuras con alguna relación se denomina clustering. En el presente script se analizará 8 datasets utilizando **k-medias** y **clasificación jerárquica**.

Funciones implementadas

Antes de realizar los correspondientes análisis a los distintos datasets, vamos a explicar brevemente las funciones implementadas que nos ayudaran a realizar lo anteriormente señalado. Para realizar algunos análisis exploratorios se usó un graficador 3D, el cual por razones de configuración no fue posible mostrarlo en el siguiente pdf.

K-medias

Es un método de agrupamiento, que tiene como objetivo la partición de un conjunto de n observaciones en k grupo.

```
#**************************************************************************
************
                                          #K-MEDIAS
#**************************************************************************
************
kmedias <- function(df, columns, k, name){
  # Aplica el algoritmo de k-medias al dataframe.
  #
  # Args:
  #   df: Dataframe que se le desea aplicar k-medias.
  #   columns: Columnas del dataframe que se le desea aplicar k-medias.
  #   k: Número de clusters que se desean.
  #   name: Nombre del dataframe.
  #
```

```
  # Returns:
  #   Retorna el modelo generado.

  modelo.kmedias = kmeans(x = df[, columns], centers = k)

  #GRAFICAMOS LOS CLUSTERS
  plot(df[, columns], col = modelo.kmedias$cluster, main = paste(c("K-MEANS:
", name)))

  # Ahora graficamos los centroides
  points(x = modelo.kmedias$centers[, columns], col = 4:8, pch = 19, cex = 3)

  return(modelo.kmedias)
}
```

Clusterización jerárquica utilizando un K para determinar la altura

Dado un número de clusters k determinar la altura requerida para que tengamos el número de cluster k.

```
#**********************************************************************
************
                                    #CLUSTERS JERARQUICOS
#**********************************************************************
************
#Dado un número de clusters k determinar la altura requerida para que
tengamos el
#número de cluster k.
clusterJD <- function(df, distancia, columns, method, k, name){
  # Aplica el algoritmo de clusterización utilizando un k para determinar
  # la altura.
  #
  # Args:
  #   df: Dataframe que se le desea aplicar k-medias.
  #   distancia: Matriz de distancias.
  #   columns: Columnas del dataframe que se le desea aplicar k-medias.
  #   method: Método deseado para aplicar el algoritmo (complete, single,
average o ward.d).
  #   k: Número de clusters que se desean.
  #   name: Nombre del dataframe.
  #
  # Returns:
  #   Retorna el modelo generado.

  #Aplicamos cluster jerarquico utilizando el metodo correspondiente
  cluster = hclust(distancia, method = method)
  ############################################################
  #Determinar la altura requerida dado un numero de clusters k
  ############################################################
  #Cortamos el dendograma con K clases
```

```
  corteD = cutree(cluster, k = k)
  #Observamos la cantidad de clusters
  unique(corteD)
  #Graficamos los clusters
  plot(df[, columns], col = corteD, main = paste(c("Cluster jerarquico K: ",
name)))

  return(corteD)
}
```

Clusterización jerárquica utilizando una medida de disimilaridad

Dada una altura h (una medida de disimilaridad) determinar el número de clusters que se obtienen.

```
#Dada una altura h (una medida de disimilaridad) determinar el número de
clusters que se obtienen.
clusterJH <- function(df, distancia, columns, method, h, name){
  # Aplica el algoritmo de clusterización utilizando un k para determinar
  # la altura.
  #
  # Args:
  #   df: Dataframe que se le desea aplicar k-medias.
  #   distancia: Matriz de distancias.
  #   columns: Columnas del dataframe que se le desea aplicar k-medias.
  #   method: Método deseado para aplicar el algoritmo (complete, single,
average o ward.d).
  #   h:  Medida de disimilaridad.
  #   name: Nombre del dataframe.
  #
  # Returns:
  #   Retorna el modelo generado.

  #Aplicamos cluster jerarquico utilizando el metodo correspondiente
  cluster = hclust(distancia, method = method)
  #Graficamos el dendogram
  plot(cluster)

  ##############################################################
  #Dada una altura h (una medida de disimilaridad) determinar
  #el numero de clusters que se obtienen
  ##############################################################
  # Cortamos por altura
  corteH = cutree(cluster, h = h)
  #Observamos la cantidad de clusters
  print(unique(corteH))
  #Graficamos los clusters
  plot(df[, columns], col = corteH, main = paste(c("Cluster jerarquico H: ",
name)))
```

```
  return(corteH)
}
```

Matriz de confusión

Es una herramienta que permite la visualización del desempeño de un algoritmo que se emplea en aprendizaje supervisado.

```
#**************************************************************************
************
#                                   MATRIZ DE CONFUSION
#**************************************************************************
************
matrizconfusion = function(class, clusters){
  x <- table(class, clusters, dnn=c("Clase", "Cluster"))
  #colnames(x) <- 0:(ncol(x)-1) #Nombres
  x1 <- x #Para buscar los maximos
  x2 <- x #Para asignar columnas
  #Tabla vacia
  for (h in 1:nrow(x1)) {
    x1[h,] <- -1
  }

  #Mientras la matriz no sea vacia.
  while (empty(x) == F) {
    maximo <- -1
    for (i in 1:nrow(x)) {
      for (j in 1:ncol(x)) {
        if (x[i,j] > maximo){
          maximo <- x[i,j]

          mi <- i
          mj <- j
        }#endif
      }#endfor
    }#endfor
    #Si no existe un valor en la diagonal.
    if (x1[mi,mi] == -1){

      x1[,mi] <- x2[,mj]
      x[,mj] <- -1
    }else{
      #Si existe, probar con otro maximo
      x[mi,mj] <- -1
    }#endif


  }#endwhile
  return(x1)
}
```

```
#Funcion que me retorna TRUE si la tabla esta vacia.
empty <- function(x){
  boolean <- T
  for (i in 1:nrow(x)) {
    for (j in 1:ncol(x)) {
      if (x[i,j] != -1){
        boolean <- F
      }
    }
  }
  return(boolean)
}
```

Precisión del modelo

La Precisión P de un modelo de predicción es la proporción del número total de predicciones que son correctas respecto al total. Se determina utilizando la ecuación: P = sum(diagonal(matriz de confusión))/ sum(todos los valores de la matriz de confusión).

```
#**************************************************************************
************
#                                    PRECISION DEL MODELO
#**************************************************************************
************
precision <- function(m){
  # Calcula la precisión del modelo utilizando la matriz de confusión.
  #
  # Args:
  #   m: Matriz de confusión.
  #
  # Returns:
  #   Retorna la precisión del modelo.

#Precision
#P = (a+d)/(a+b+c+d)
  return(sum(diag(m)) /sum(m))
}
```

Mejor modelo

Selecciona el mejor modelo utilizando como criterio la mayor precisión.

```
#**************************************************************************
************
#                                MEJOR MODELO SEGUN LA PRECISION
#**************************************************************************
************
bestmodel <- function(x){
  # Busca el mejor modelo.
```

```
  #
  # Args:
  #   x: Modelo con mayor precisión.
  #
  # Returns:
  #   Retorna el mejor modelo.


  if (x == 1){
    return("K-MEDIAS")
  }else if (x == 2){
    return("CLASIFICACION JERARQUICA K: METHOD COMPLETE")
  }else if (x == 3){
    return("CLASIFICACION JERARQUICA H: METHOD COMPLETE")
  }else if (x == 4){
    return("CLASIFICACION JERARQUICA K: METHOD SINGLE")
  }else if (x == 5){
    return("CLASIFICACION JERARQUICA H: METHOD SINGLE")
  }else if (x == 6){
    return("CLASIFICACION JERARQUICA K: METHOD AVERAGE")
  }else if (x == 7){
    return("CLASIFICACION JERARQUICA H: METHOD AVERAGE")
  }else if (x == 8){
    return("CLASIFICACION JERARQUICA K: METHOD WARD.D")
  }else{
    return("CLASIFICACION JERARQUICA H: METHOD WARD.D")
  }
}
```

Encontrando estructuras

A continuación aplicaremos los siguientes objetivos a los datasets correspondientes:

- Usar métodos exploratorios.
- Elaborar soluciones a problemas específcos de cada dataset.
- Usar herramientas de clustering.

a.csv

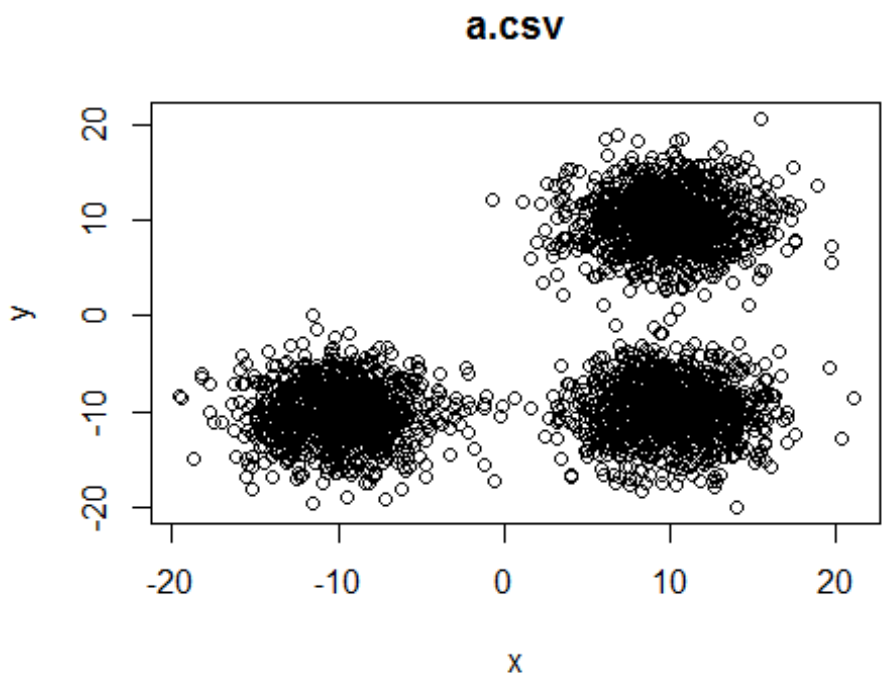
- Preprocesamiento:

```
setwd("C:/Users/Eric/Desktop/AprendizajeNoSupervisado/")
name = "a.csv"
#Lectura de datos.
df = read.csv(file = "data/a.csv", header = F)
#Modificamos el nombre de las columnas por comodidad.
colnames(df) <- c("x","y","class")
#Coloco las clases del 1:n
df$class = as.numeric(df$class)
```

```r
if (min(df$class) == 0){
  df$class <- df$class + 1
}

#Ordenamos la columna clase
df <- df[ order(df$class), ]
```

- Analisis exploratorio del dataset:

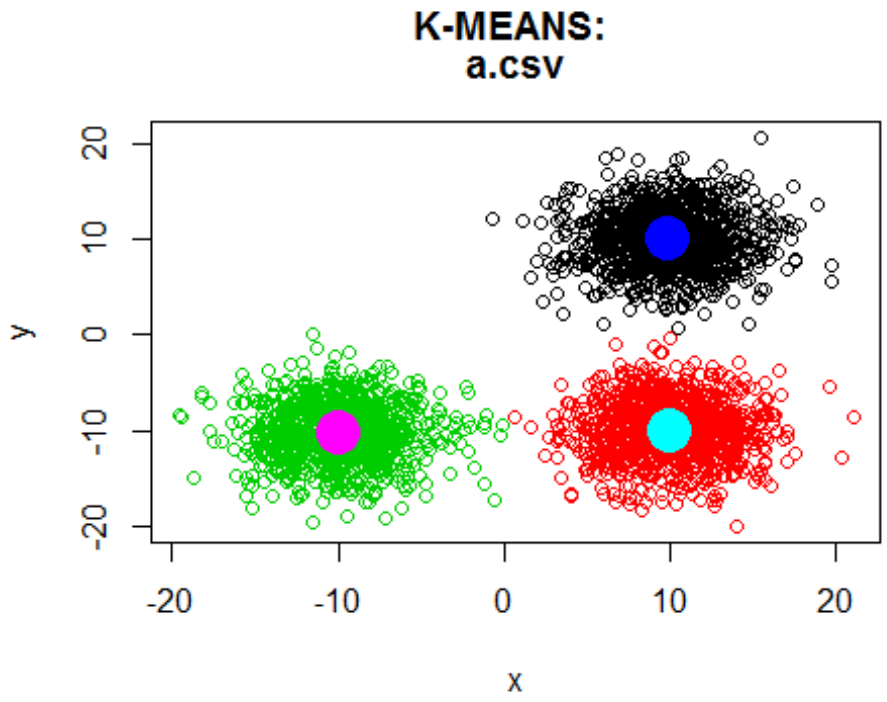
```r
#*************************************************************************
************
                              #Analisis exploratorio del dataset
#*************************************************************************
************
#Podemos observar que hay 3 columnas.
head(df)

##            x         y class
## 2  13.055035  9.445506     1
## 9  10.125790  8.255876     1
## 18 10.892337  8.471922     1
## 25  7.922571  9.556655     1
## 26  8.694283 12.361170     1
## 27  7.683897  7.856547     1

#Observamos cuantos elementos hay de cada clase.
table(df$class)

##
##    1    2    3
## 1000 1000 1000

#1     2    3
#1000  1000 1000

#Grafico
plot(df$x, df$y, xlab = "x", ylab = "y", main = name)
```




```
#Podemos observar 3 conglomerados

length(unique(df$class))

## [1] 3

#Existen 3 clases.
```

K-medias:

```
#*************************************************************************
************
                                        #K-MEDIAS
#*************************************************************************
************
#Aplicamos k=3 ya que identificamos 3 conglomerados y existen 3 clases.
#kmedias(Dataframe, Columnas, K, name)
modeloK <- kmedias(df, 1:2, 3, name)
```



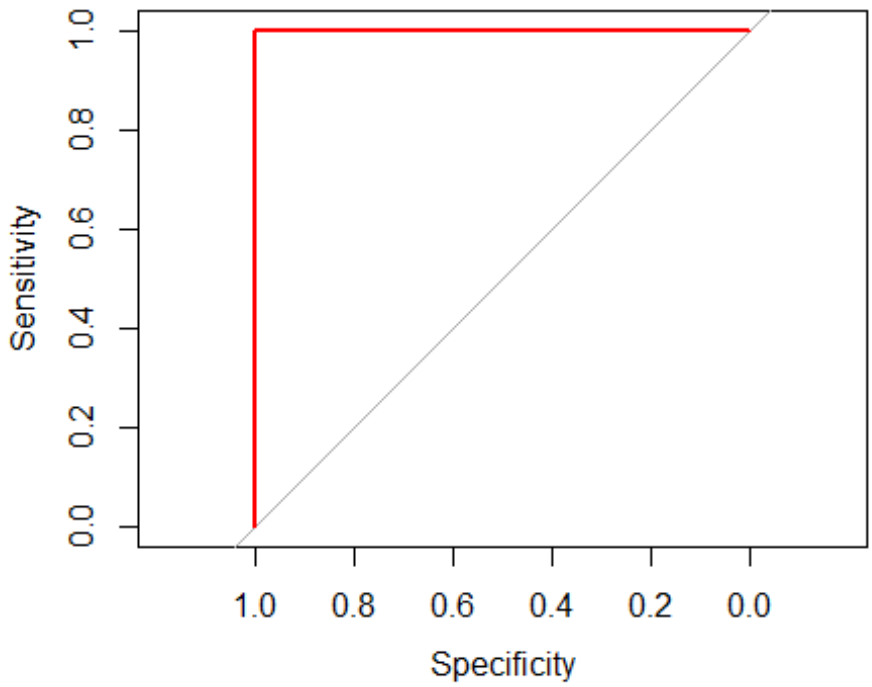

```
#Generamos la matriz de confusion
MatrixConfusionK <- matrizconfusion(df$class,modeloK$cluster)
MatrixConfusionK
```

```
##       Cluster
## Clase    1    2    3
##     1 1000    0    0
##     2    0 1000    0
##     3    1    0  999
```

```
#Calculamos la precision del modelo
PrecisionK <- precision(MatrixConfusionK)
PrecisionK
```

```
## [1] 0.9996667
```

```
#Generamos la curva de ROC
modeloKROC <- roc(df$class, modeloK$cluster)
plot(modeloKROC,type="l",col="red")
```



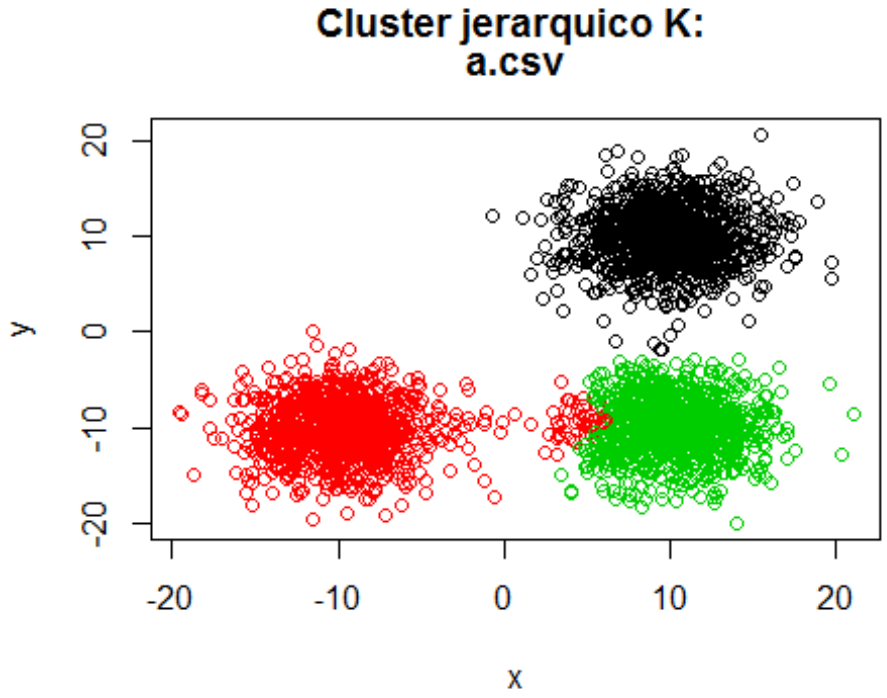

```
##
## Call:
## roc.default(response = df$class, predictor = modeloK$cluster)
##
## Data: modeloK$cluster in 1000 controls (df$class 1) < 1000 cases (df$class
2).
## Area under the curve: 1
```

Clusters jerárquicos:

```
#*************************************************************************
************
                                    #CLUSTER JERARQUICOS
#*************************************************************************
************
#Copy del dataset
datos = df
#Elimino la columna clase para realizar aprendizaje no supervisado.
datos$class <- NULL
#Convierto el dataframe en una matriz
datos= as.matrix(datos)
#Calculamos la matriz de distancia
distancia = dist(datos)
```

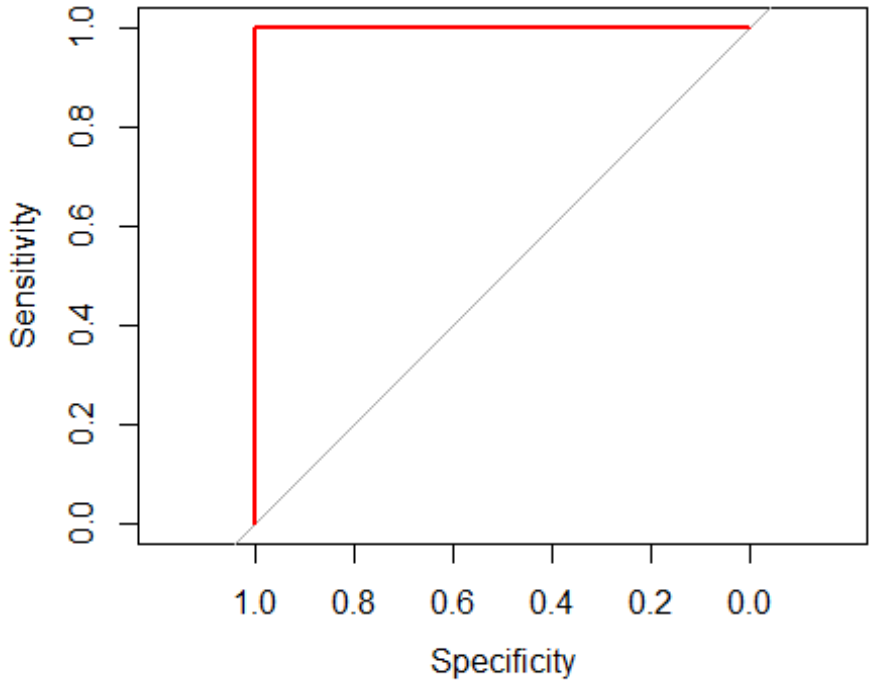
Método complete:

```
#-------------------------------------------------------------------------
----------
                                        #METHOD COMPLETE
#-------------------------------------------------------------------------
----------
#Dado un numero de clusters k determinar la altura requerida
#para que tengamos el numero de cluster k.
clustersD <- clusterJD(df, distancia, 1:2, "complete", 3, name)
```



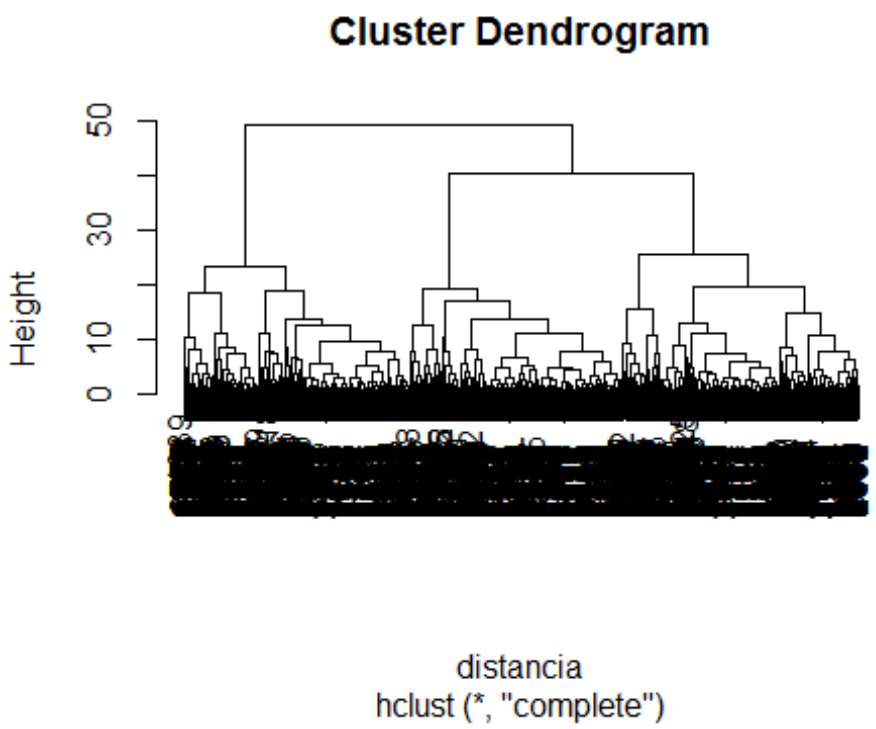

```
#Generamos la matriz de confusion
MatrixConfusionCJDC <- matrizconfusion(df$class, clustersD)
MatrixConfusionCJDC

##      Cluster
## Clase    1    2    3
##     1 1000    0    0
##     2    0 1000    0
##     3    6   47  947

#Calculamos la precision del modelo
PrecisionDC <- precision(MatrixConfusionCJDC)
PrecisionDC

## [1] 0.9823333
```

```r
#Generamos la curva de ROC
modeloDC <- roc(df$class, clustersD)
plot(modeloDC,type="l",col="red")
```

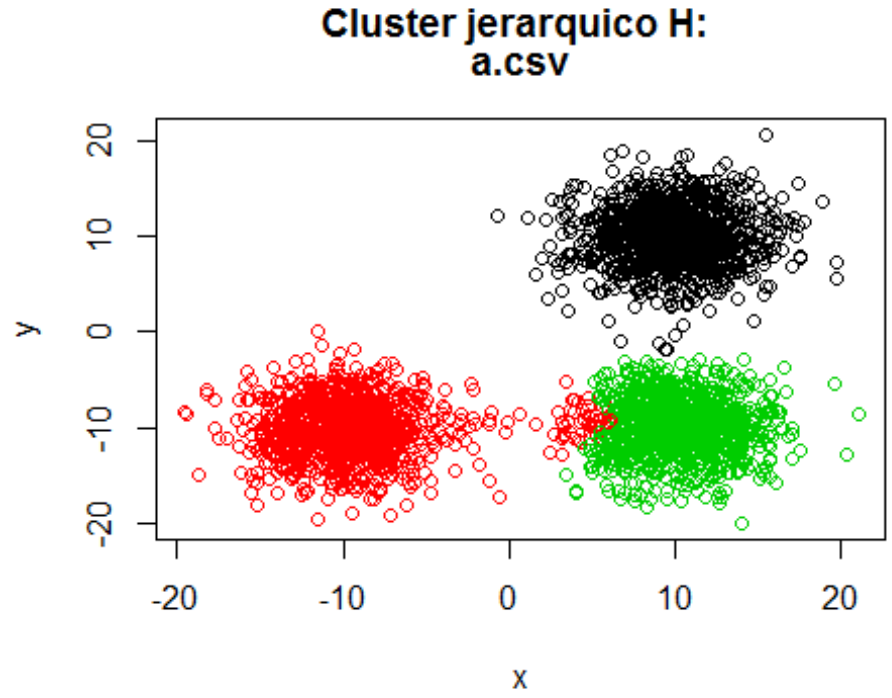



```
##
## Call:
## roc.default(response = df$class, predictor = clustersD)
##
## Data: clustersD in 1000 controls (df$class 1) < 1000 cases (df$class 2).
## Area under the curve: 1
```

```r
#*********************************************************
#Dada una altura h (una medida de disimilaridad) determinar
#el numero de clusters que se obtienen.
clustersH <- clusterJH(df, distancia, 1:2, "complete", 30, name)
```


Cluster Dendrogram
Height
0
10
30
50
distancia
hclust (*, "complete")

```
## [1] 1 2 3
```


Cluster jerarquico H:
a.csv
y
-20
-10
0
10
20
x

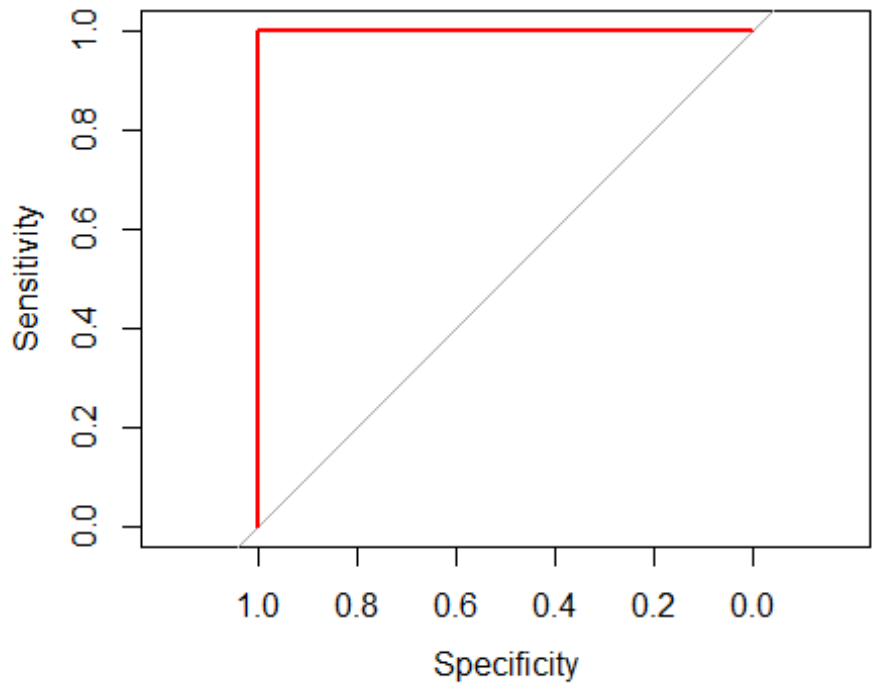
```r
#Generamos la matriz de confusion
MatrixConfusionCJHC <- matrizconfusion(df$class, clustersH)
MatrixConfusionCJHC
```

```
##      Cluster
## Clase    1    2    3
##     1 1000    0    0
##     2    0 1000    0
##     3    6   47  947
```

```r
#Calculamos la precision del modelo
PrecisionHC <- precision(MatrixConfusionCJHC)
PrecisionHC
```

```
## [1] 0.9823333
```

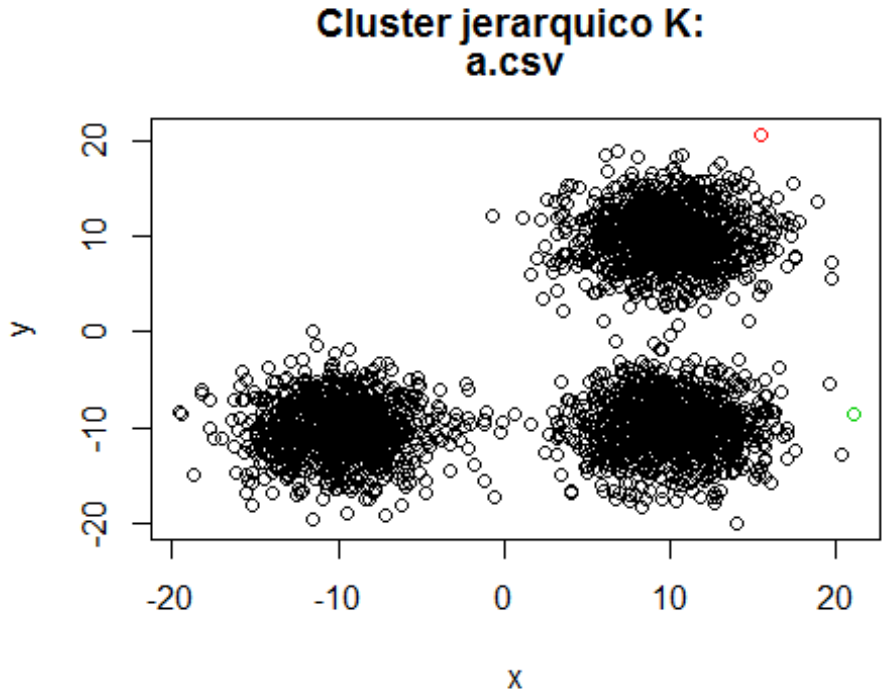
```r
#Generamos la curva de ROC
modeloHC <- roc(df$class, clustersH)
plot(modeloHC,type="l",col="red")
```




```
##
## Call:
## roc.default(response = df$class, predictor = clustersH)
##
## Data: clustersH in 1000 controls (df$class 1) < 1000 cases (df$class 2).
## Area under the curve: 1
```

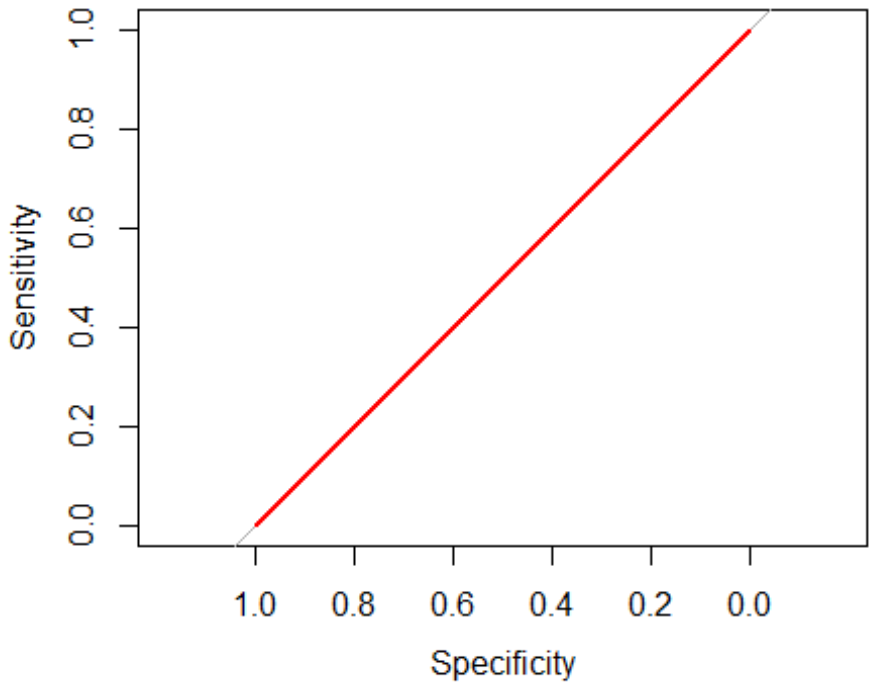
Método single:

```
#-------------------------------------------------------------------------
----------
                                        #METHOD SINGLE
#-------------------------------------------------------------------------
----------
#Dado un numero de clusters k determinar la altura requerida
#para que tengamos el numero de cluster k.
clustersD <- clusterJD(df, distancia, 1:2, "single", 3, name)
```



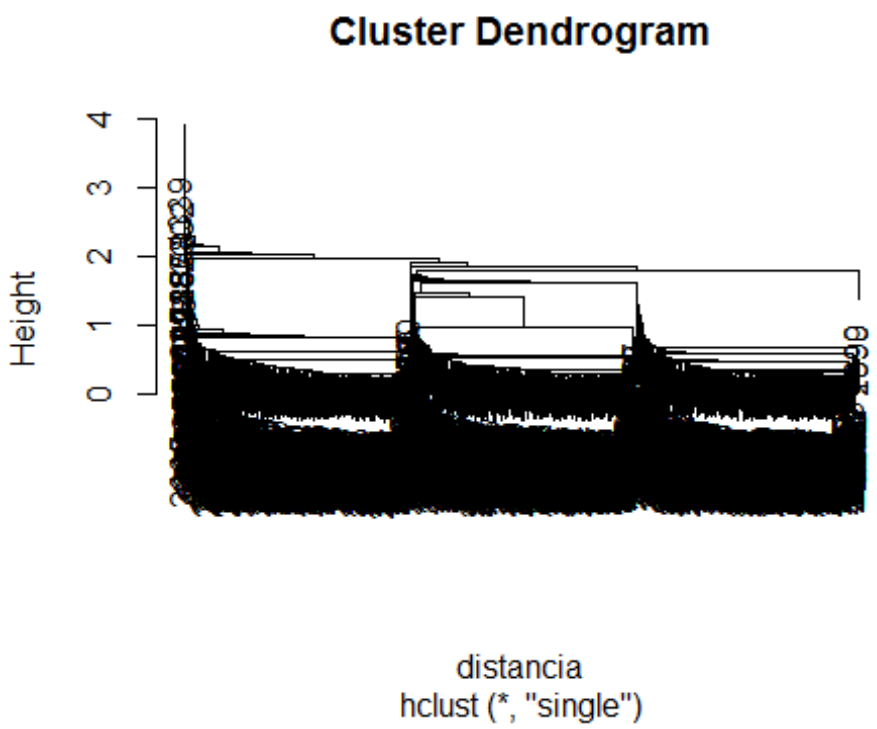

```
#Generamos la matriz de confusion
MatrixConfusionCJDS <- matrizconfusion(df$class, clustersD)
MatrixConfusionCJDS

##      Cluster
## Clase    1    2    3
##     1    1  999    0
##     2    0 1000    0
##     3    0  999    1

#Calculamos la precision del modelo
PrecisionDS <- precision(MatrixConfusionCJDS)
PrecisionDS

## [1] 0.334
```

```r
#Generamos la curva de ROC
modeloDS <- roc(df$class, clustersD)
plot(modeloDS,type="l",col="red")
```

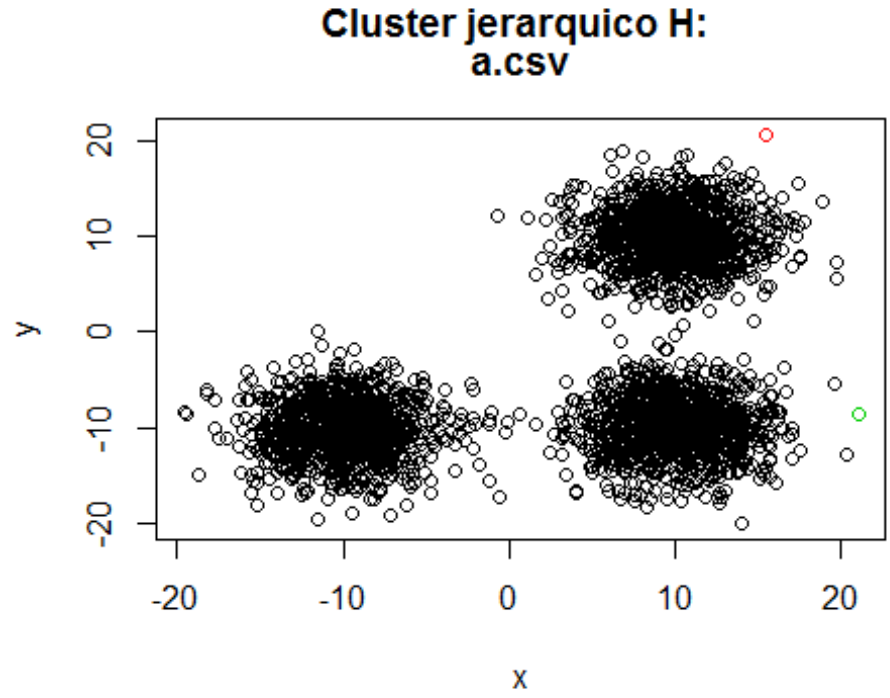



```
##
## Call:
## roc.default(response = df$class, predictor = clustersD)
##
## Data: clustersD in 1000 controls (df$class 1) < 1000 cases (df$class 2).
## Area under the curve: 0.4995
```

```r
#*********************************************************
#Dada una altura h (una medida de disimilaridad) determinar
#el numero de clusters que se obtienen.
clustersH <- clusterJH(df, distancia, 1:2, "single", 3.5, name)
```


Cluster Dendrogram
Height
0
1
2
3
4
distancia
hclust (*, "single")

```
## [1] 1 2 3
```


Cluster jerarquico H:
a.csv
y
20
10
0
-10
-20
-20
-10
0
10
20
x

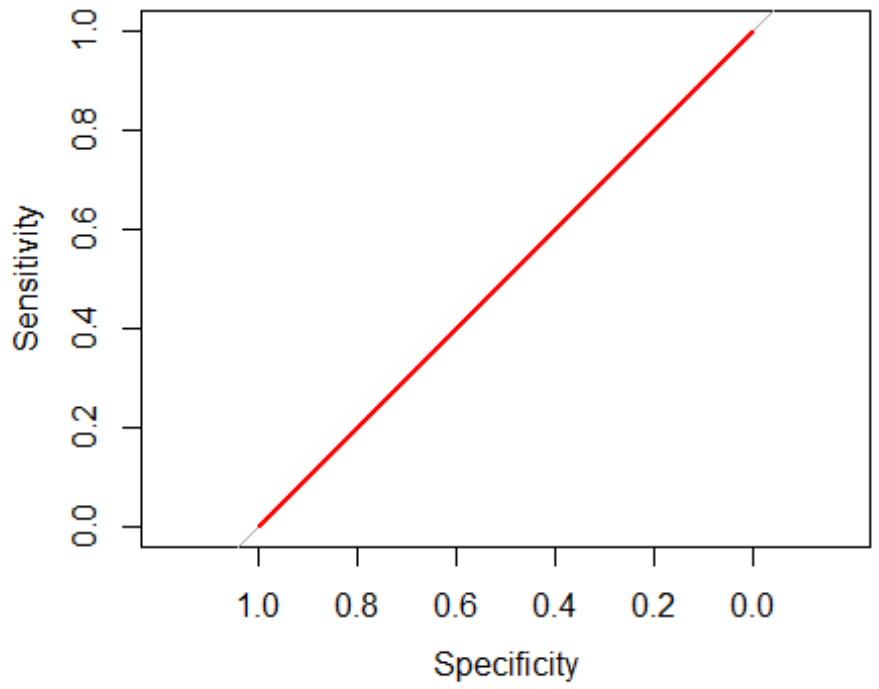
```r
#Generamos la matriz de confusion
MatrixConfusionCJHS <- matrizconfusion(df$class, clustersH)
MatrixConfusionCJHS
```

```
##       Cluster
## Clase    1    2    3
##     1    1  999    0
##     2    0 1000    0
##     3    0  999    1
```

```r
#Calculamos la precision del modelo
PrecisionHS <- precision(MatrixConfusionCJHS)
PrecisionHS
```

```
## [1] 0.334
```

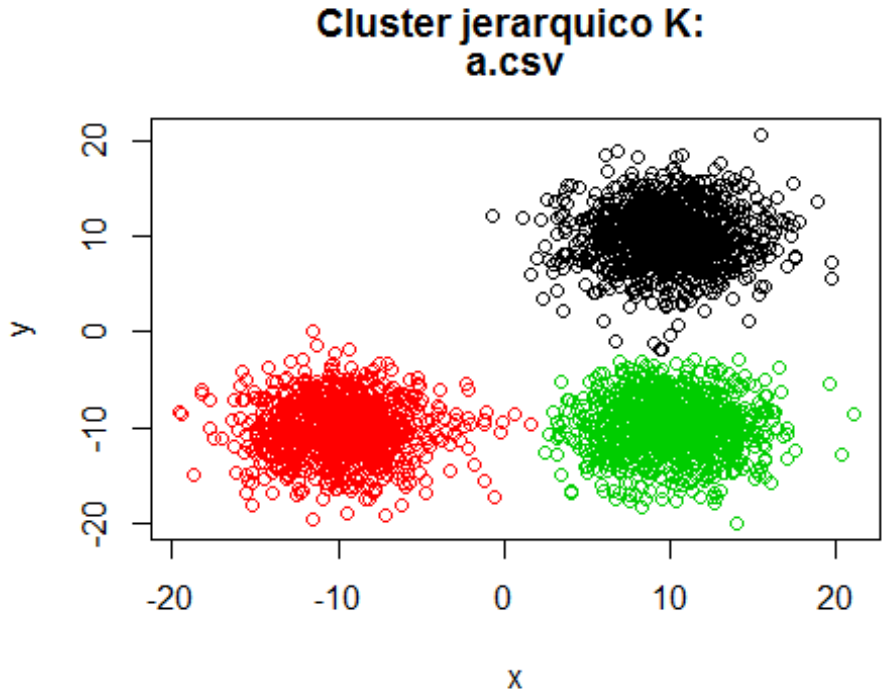
```r
#Generamos la curva de ROC
modeloHS <- roc(df$class, clustersH)
plot(modeloHS,type="l",col="red")
```




```
##
## Call:
## roc.default(response = df$class, predictor = clustersH)
##
## Data: clustersH in 1000 controls (df$class 1) < 1000 cases (df$class 2).
## Area under the curve: 0.4995
```

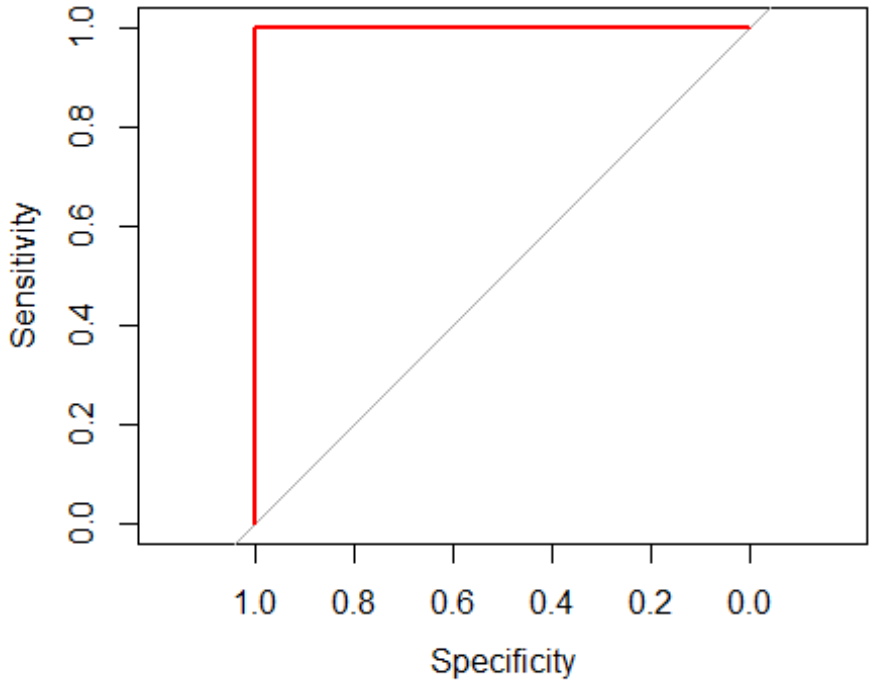
Método average:

```
#-------------------------------------------------------------------------
----------
                                        #METHOD AVERAGE
#-------------------------------------------------------------------------
----------
#Dado un numero de clusters k determinar la altura requerida
#para que tengamos el numero de cluster k.
clustersD <- clusterJD(df, distancia, 1:2, "average", 3, name)
```



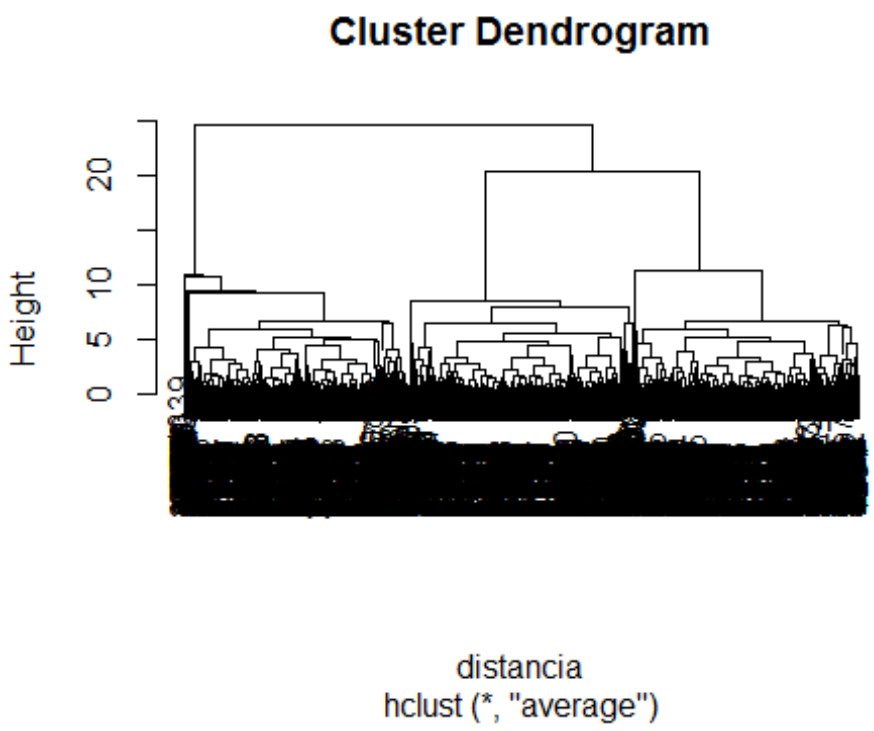

```
#Generamos la matriz de confusion
MatrixConfusionCJDA <- matrizconfusion(df$class, clustersD)
MatrixConfusionCJDA

##      Cluster
## Clase    1    2    3
##     1 1000    0    0
##     2    0 1000    0
##     3    6    2  992

#Calculamos la precision del modelo
PrecisionDA <- precision(MatrixConfusionCJDA)
PrecisionDA

## [1] 0.9973333
```

```r
#Generamos la curva de ROC
modeloDA <- roc(df$class, clustersD)
plot(modeloDA,type="l",col="red")
```

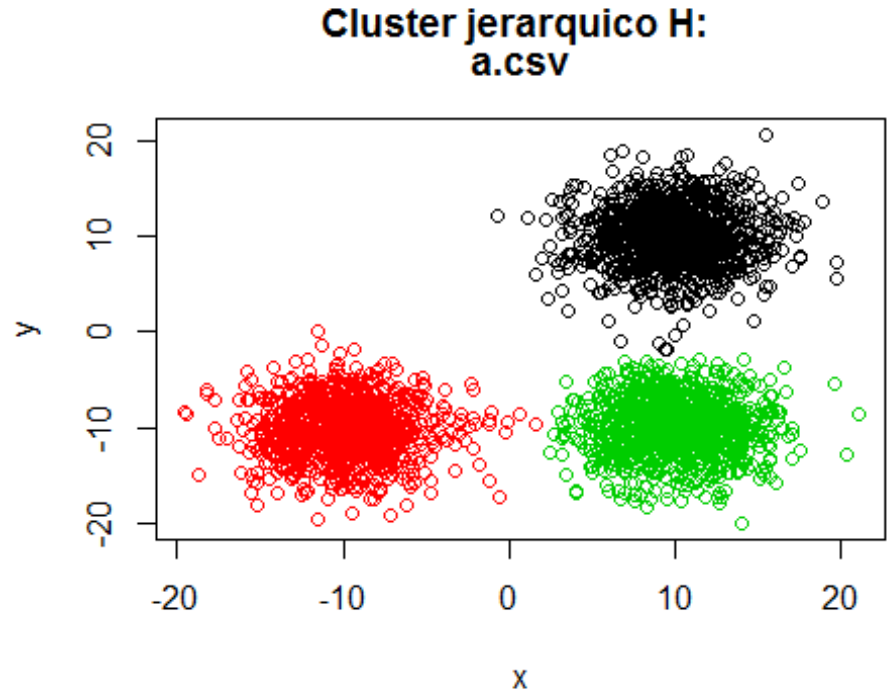



```
##
## Call:
## roc.default(response = df$class, predictor = clustersD)
##
## Data: clustersD in 1000 controls (df$class 1) < 1000 cases (df$class 2).
## Area under the curve: 1
```

```r
#*********************************************************
#Dada una altura h (una medida de disimilaridad) determinar
#el numero de clusters que se obtienen.
clustersH <- clusterJH(df, distancia, 1:2, "average", 15, name)
```


Cluster Dendrogram
Height
0
5
10
20
distancia
hclust (*, "average")

```
## [1] 1 2 3
```


Cluster jerarquico H:
a.csv
y
-20
-10
0
10
20
x

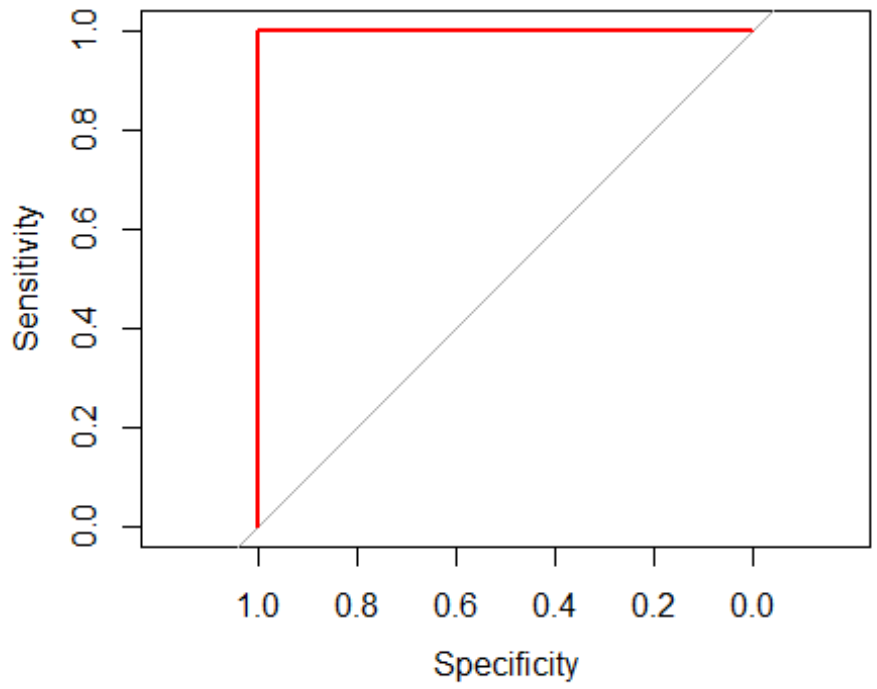
```r
#Generamos la matriz de confusion
MatrixConfusionCJHA <- matrizconfusion(df$class, clustersH)
MatrixConfusionCJHA
```

```
##       Cluster
## Clase    1    2    3
##     1 1000    0    0
##     2    0 1000    0
##     3    6    2  992
```

```r
#Calculamos la precision del modelo
PrecisionHA <- precision(MatrixConfusionCJHA)
PrecisionHA
```

```
## [1] 0.9973333
```

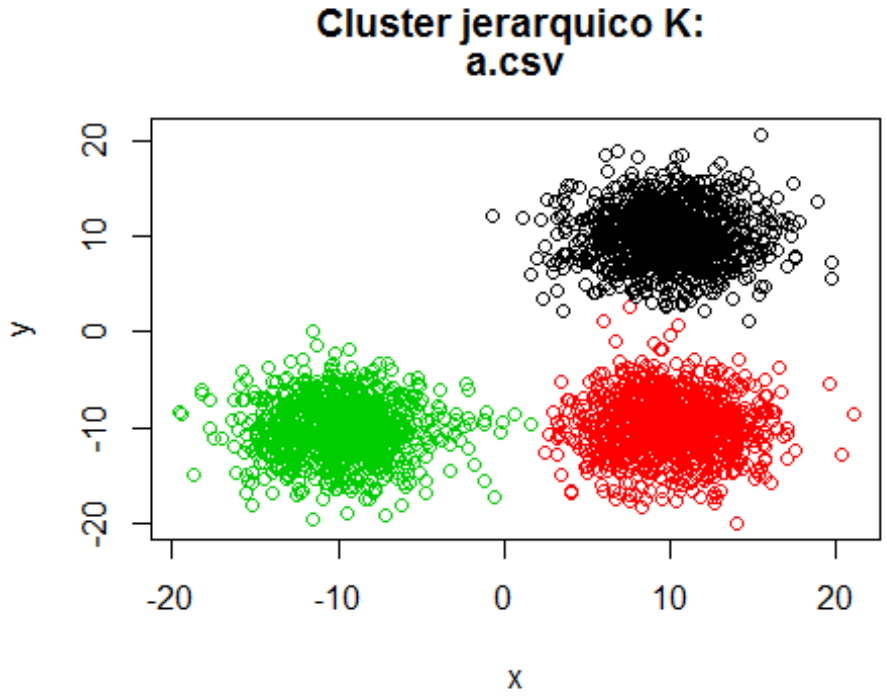
```r
#Generamos la curva de ROC
modeloHA <- roc(df$class, clustersH)
plot(modeloHA,type="l",col="red")
```




```
##
## Call:
## roc.default(response = df$class, predictor = clustersH)
##
## Data: clustersH in 1000 controls (df$class 1) < 1000 cases (df$class 2).
## Area under the curve: 1
```

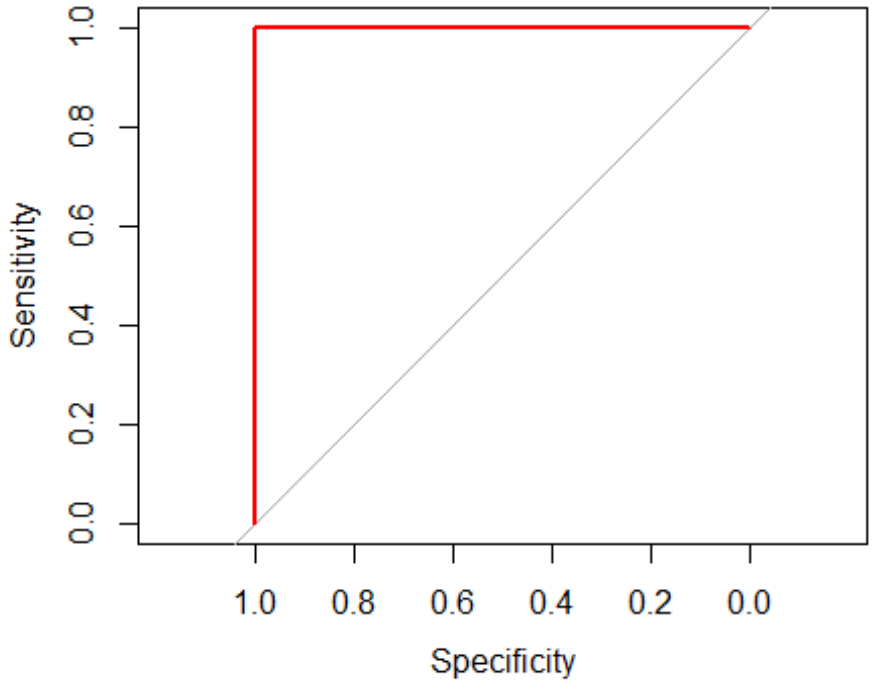
Método ward.D

```
#-------------------------------------------------------------------------
----------
#METHOD ward.D
#-------------------------------------------------------------------------
----------
#Dado un numero de clusters k determinar la altura requerida
#para que tengamos el numero de cluster k.
clustersD <- clusterJD(df, distancia, 1:2, "ward.D", 3, name)
```



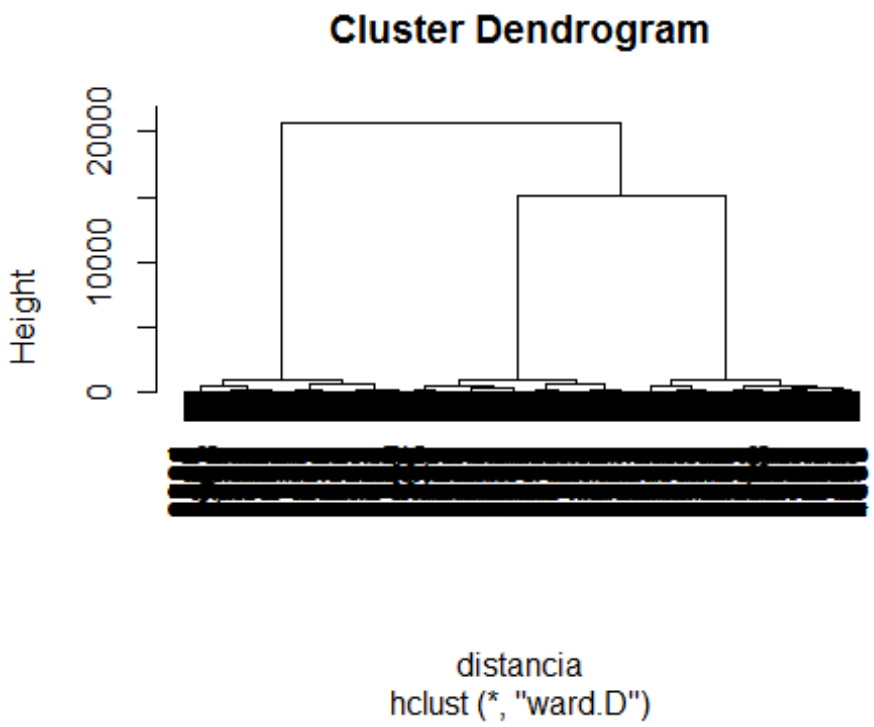

```
#Generamos la matriz de confusion
MatrixConfusionCJDW <- matrizconfusion(df$class, clustersD)
MatrixConfusionCJDW

##      Cluster
## Clase    1    2    3
##     1  998    0    2
##     2    0 1000    0
##     3    0    2  998

#Calculamos la precision del modelo
PrecisionDW <- precision(MatrixConfusionCJDW)
PrecisionDW

## [1] 0.9986667
```

```r
#Generamos la curva de ROC
modeloDW <- roc(df$class, clustersD)
plot(modeloDW,type="l",col="red")
```

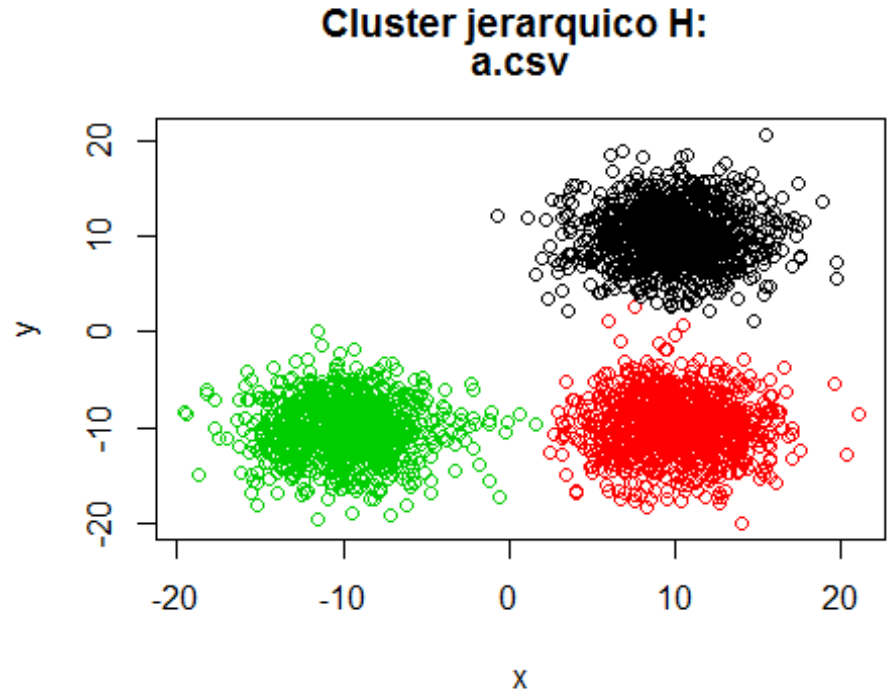



```
##
## Call:
## roc.default(response = df$class, predictor = clustersD)
##
## Data: clustersD in 1000 controls (df$class 1) < 1000 cases (df$class 2).
## Area under the curve: 1
```

```r
#*********************************************************
#Dada una altura h (una medida de disimilaridad) determinar
#el numero de clusters que se obtienen.
clustersH <- clusterJH(df, distancia, 1:2, "ward.D", 10000, name)
```


Cluster Dendrogram
Height
0
10000
20000
distancia
hclust (*, "ward.D")

```
## [1] 1 2 3
```


Cluster jerarquico H:
a.csv
y
-20
-10
0
10
20
x

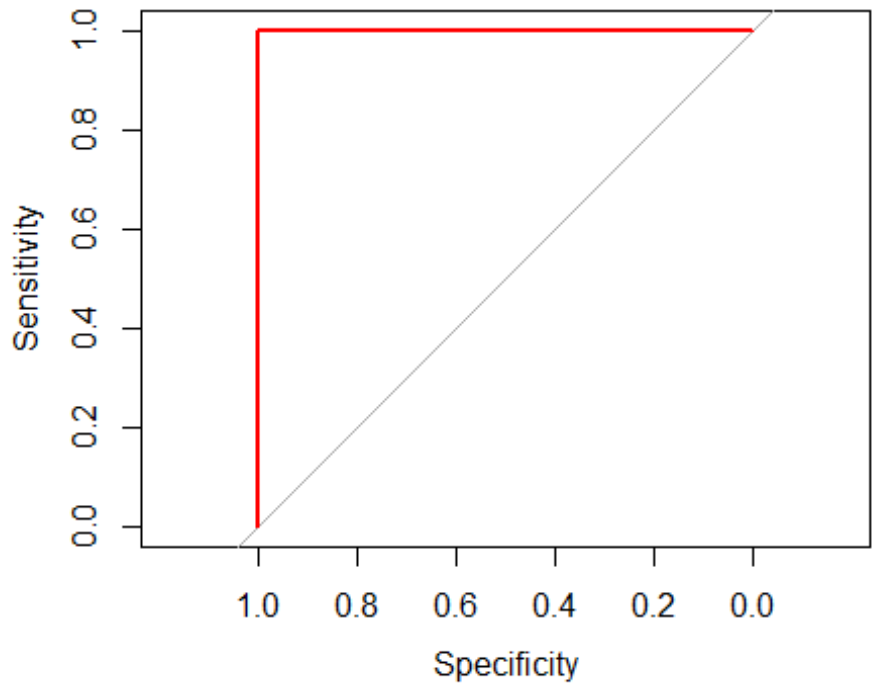
```r
#Generamos la matriz de confusion
MatrixConfusionCJHW <- matrizconfusion(df$class, clustersH)
MatrixConfusionCJHW
```

```
##       Cluster
## Clase    1    2    3
##     1  998    0    2
##     2    0 1000    0
##     3    0    2  998
```

```r
#Calculamos la precision del modelo
PrecisionHW <- precision(MatrixConfusionCJHW)
PrecisionHW
```

```
## [1] 0.9986667
```

```r
#Generamos la curva de ROC
modeloHW <- roc(df$class, clustersH)
plot(modeloHW,type="l",col="red")
```




```
##
## Call:
## roc.default(response = df$class, predictor = clustersH)
##
## Data: clustersH in 1000 controls (df$class 1) < 1000 cases (df$class 2).
## Area under the curve: 1
```

Mejor modelo

```
#-------------------------------------------------------------------------
----------
#                                   MEJOR MODELO
#-------------------------------------------------------------------------
----------
precisiones <- c(PrecisionK, PrecisionDC, PrecisionHC, PrecisionDS,
PrecisionHS, PrecisionDA,
                 PrecisionHA, PrecisionDW, PrecisionHW)
x <- which.max(precisiones)
mejormodelo <- bestmodel(x)
cat("El mejor modelo es: ", mejormodelo, ", que posee una precision de: ",
precisiones[x], ".")

## El mejor modelo es:  K-MEDIAS , que posee una precision de:  0.9996667 .
```

a_big.csv

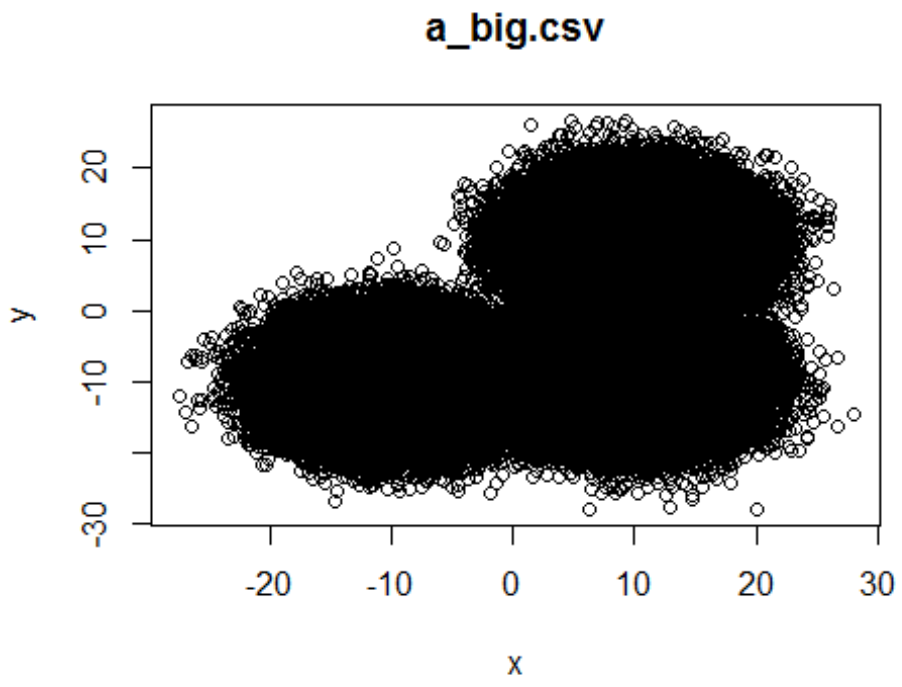
- Preprocesamiento:

```
setwd("C:/Users/Eric/Desktop/AprendizajeNoSupervisado/")
name = "a_big.csv"
#Lectura de datos.
df = read.csv(file = "data/a_big.csv", header = F)
#Modificamos el nombre de las columnas por comodidad.
colnames(df) <- c("x","y","class")
#Coloco las clases del 1:n
df$class = as.numeric(df$class)
if (min(df$class) == 0){
  df$class <- df$class + 1
}
```

- Analisis exploratorio del dataset:

```
#*************************************************************************
************
#Analisis exploratorio del dataset
#*************************************************************************
************
#Podemos observar que hay 3 columnas.
head(df)

##           x         y class
## 1  8.694977 -7.254259     3
## 2 11.537866  9.091760     1
## 3 -6.672032 -8.277564     2
## 4 11.003768  9.930918     1
## 5  8.762679  1.518045     1
## 6 10.325578 14.677636     1

#Observamos cuantos elementos hay de cada clase.
table(df$class)
```

```
##
##      1      2      3
## 100000 100000 100000

#1       2        3
#100000 100000  100000

#Grafico
plot(df$x, df$y, xlab = "x", ylab = "y", main = name)
```

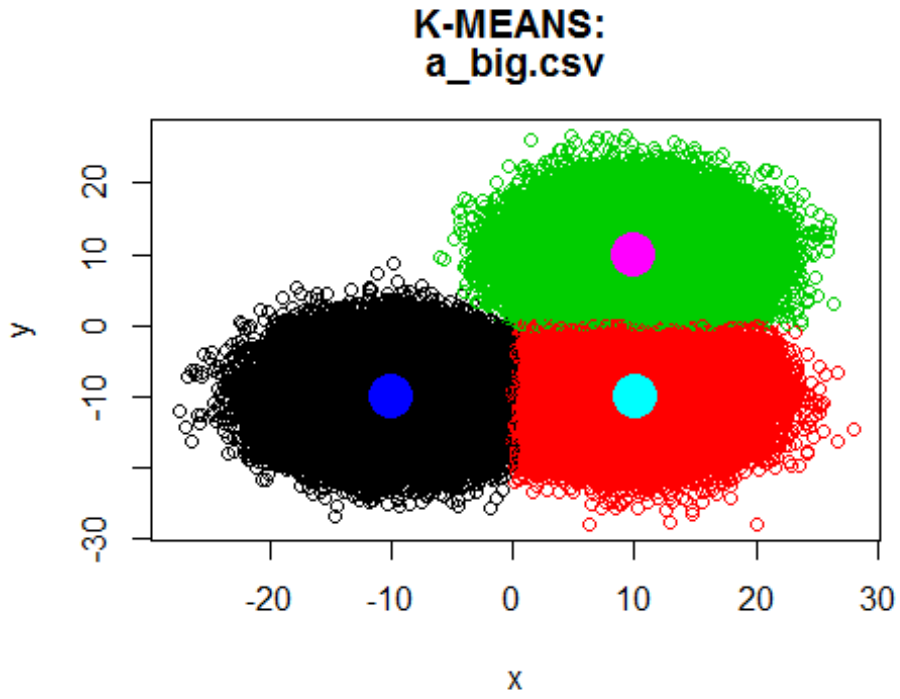



```
#Podemos observar 3 conglomerados

length(unique(df$class))

## [1] 3

#Existen 3 clases.
```

K-medias:

```
#*************************************************************************
***********
                                          #K-MEDIAS
#*************************************************************************
***********
#Aplicamos k=3 ya que identificamos 3 conglomerados y existen 3 clases.
```

```r
#kmedias(Dataframe, Columnas, K, name)
modeloK <- kmedias(df, 1:2, 3, name)
```



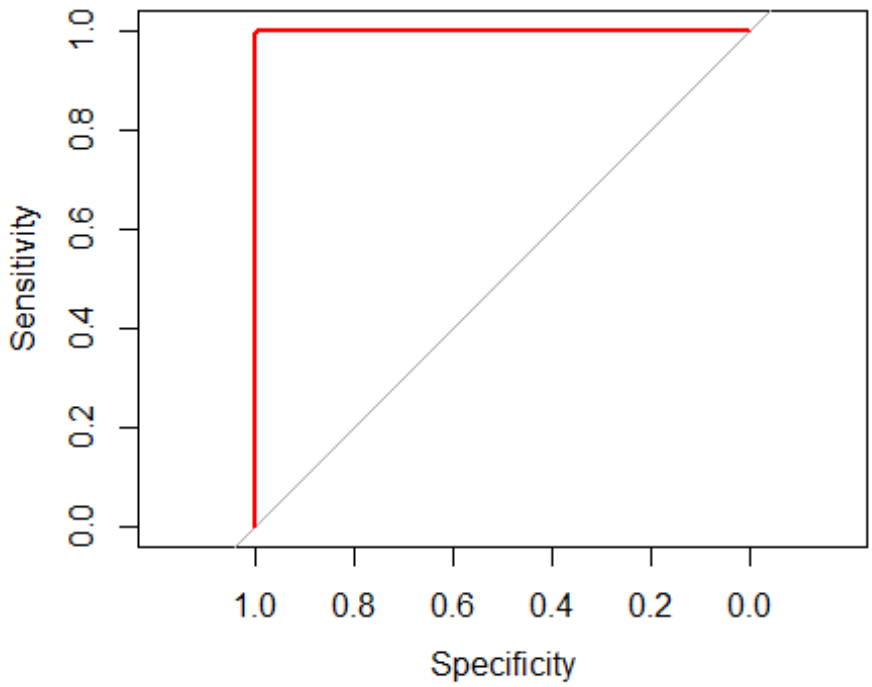

```r
#Generamos la matriz de confusion
MatrixConfusion <- matrizconfusion(df$class,modeloK$cluster)
MatrixConfusion
```

```
##       Cluster
## Clase     1     2     3
##     1 99405    11   584
##     2    14 99401   585
##     3   666   589 98745
```

```r
#Calculamos la precision del modelo
PrecisionK <- precision(MatrixConfusionK)
PrecisionK
```

```
## [1] 0.9996667
```

```r
#Generamos la curva de ROC
modeloKROC <- roc(df$class, modeloK$cluster)
plot(modeloKROC,type="l",col="red")
```




```
##
## Call:
## roc.default(response = df$class, predictor = modeloK$cluster)
##
## Data: modeloK$cluster in 100000 controls (df$class 1) > 100000 cases
(df$class 2).
## Area under the curve: 0.9999
```

Clusters jerárquicos:

Debido a que la matriz de distancia es demasiado grande y no cabe en memoria, no podemos realizar clusterización jerárquica.

Implementación K-medias:

En primer lugar implemente un algoritmo de k-medias que utiliza como medida de distancia la euclidiana con norma 2, en segundo lugar los centroides iniciales de a_big.csv son los centroides resultantes de aplicarle k-medias a un sample estratificado del conjunto original.

```
#**************************************************************************
***********
                                  #Implementacion k-medias
#**************************************************************************
```

```
************
#*************************************************************************
****
#                                        K-MEDIAS
#*************************************************************************
****
kkmedias <- function(x, centers, dist, maxIter) {
  # Aplica el algoritmo de k-medias al dataframe.
  #
  # Args:
  #   df: Dataframe que se le desea aplicar k-medias.
  #   centers: Centroides.
  #   dist: Matriz de distancias
  #   maxIter: Numero maximo de iteracciones.
  #
  # Returns:
  #   Retorna una lista con los clusters generados por el modelo y los
centroides finales.

  clusters <- vector(maxIter, mode="list")
  centers <- vector(maxIter, mode="list")

    for(i in 1:maxIter) {
      distsToCenters <- dist
      cluster <- apply(distsToCenters, 1, which.min)
      center <- apply(x, 2, tapply, cluster, mean)
      clusters <- cluster
      centers <- center
    }#endfor

    return(list(clusters = clusters, centers= centers))
}
```

Cálculo de la distancia euclidiana:

```
#*************************************************************************
****
#                              DISTANCIA EUCLIDIANA
#*************************************************************************
****
distancia <- function(df, centros) {
  # Calcula la distancia entre los centros iniciales
  # y todos los puntos del dataframe utilizando la distancia euclidiana de
norma 2.
  #
  # Args:
  #   df: Puntos del dataframe.
  #   centros: Centros iniciales.
  #
  # Returns:
```

```
  #   Retorna todas las distancias de los centros con los del dataframe.

  #Genero la matriz de las distancias vacia.
  euclidiana <- matrix(NA,
                       nrow=dim(df)[1],
                       ncol=dim(centros)[1])

  #Calculo la distancia euclidiana con cada centro.
  for(i in 1:nrow(centros)) {
    #Distancia euclidiana: d_E(P_1,P_2)=\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}
    euclidiana[,i] <- sqrt(rowSums(t(t(df) - centros[i,])^2))
  }#endfor

  return(euclidiana)
}
```

La siguiente función genera un sample estratificado con ayuda de la librería **"sampling"** de R.

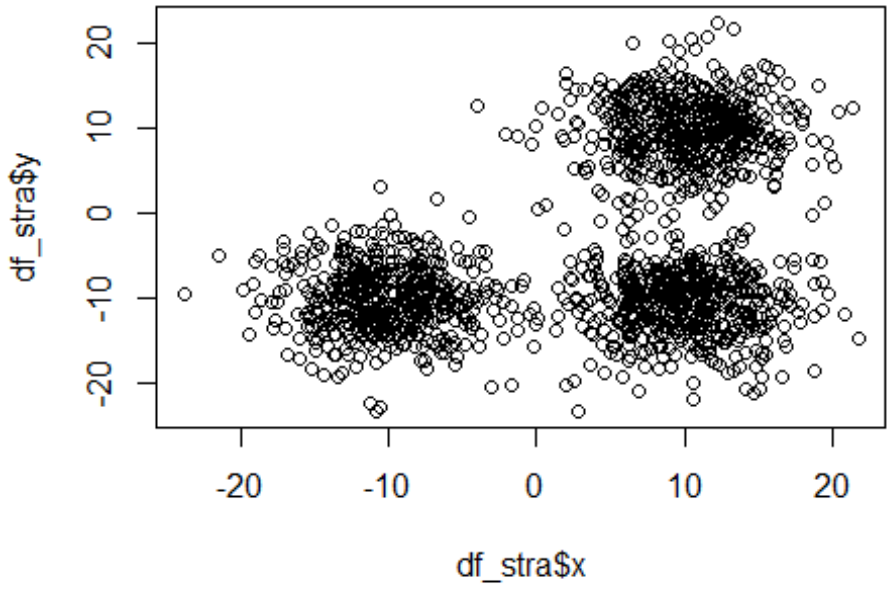
```
#*********************************************************************
****
#                              SAMPLE ESTRATIFICADO
#*********************************************************************
****
stratified <- function(df, class, size) {
  # Genera un un sample estratificado del dataset.
  #
  # Args:
  #   df: Dataframe original.
  #   class: Columna clase.
  #   size: Tamano del dataframe resultado
  #
  # Returns:
  #   Retorna un dataframe estratificado.

  temp <- df[order(df[class]),]
  if (size < 1) {
    size = ceiling(table(temp[class]) * size)
  } else if (size >= 1) {
    size = rep(size, times=length(table(temp[class])))
  }
  strat <- strata(temp, stratanames = names(temp[class]),
                  size = size, method = "srswor")
  (dsample <- getdata(temp, strat))
}
```

A partir del subconjunto de **a_big.csv** calculo los centroides que se utilizarán como centroides iniciales.

```
#*************************************************************************
****
#                       CALCULO DE CENTROIDES SOBRE EL SUBCONJUNTO
#*************************************************************************
****
#Genero un sample estratificado de size 500.
df_stra <- stratified(df, 3, 500)
```

Podemos observar que se parece a.csv.

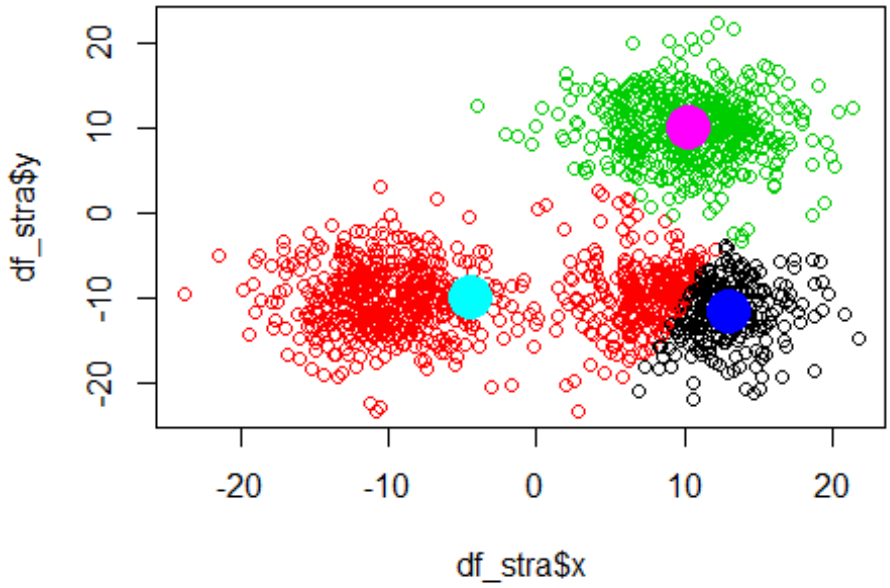
```
#Podemos observar que se parece a.csv
plot(df_stra$x,df_stra$y)
```




```
stra <- df_stra
stra$class <- NULL
stra$ID_unit <- NULL
stra$Prob <- NULL
stra$Stratum <- NULL
kstra <- as.matrix(stra)
#Genero centroides aleatorios
centers <- kstra[sample(nrow(kstra), 3),]

#Calculo la distancia euclidiana
dist <- distancia(kstra, centers)
```

De esta forma queda la clusterización del subconjunto.

```r
#Genero el k-medias para el sample.
res <- kkmedias(kstra, centers, dist, 1000)
plot(df_stra$x, df_stra$y, col = res$clusters)

# Ahora graficamos los centroides
points(x = res$centers, col = 4:8, pch = 19, cex = 3)
```



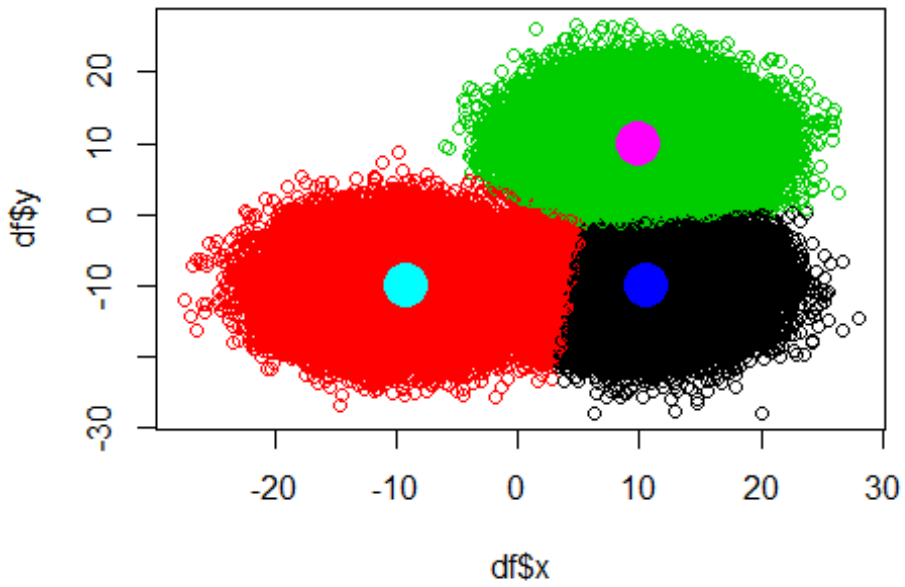
Ahora utilizando los centroides anteriores, los utilizaremos como centroides iniciales de **a_big.csv**

```r
#*************************************************************************
****
#APLICAR K-MEDIAS UTILIZANDO TODO EL CONJUNTO DE DATOS CON  LOS CENTROIDES
#FINALES DEL SAMPLE ESTRATIFICADO
#*************************************************************************
****
dfcompleto <- df
dfcompleto$class <- NULL
dfcompleto <- as.matrix(dfcompleto)

#Utilizo como centroides iniciales, los centroides finales del sampling.
centers <- res$centers

dist <- distancia(dfcompleto, centers)

#Coloco un maximo de iteracciones bajo ya que indicandole unos centroides
iniciales
#adecuados, el algoritmo converge rapido.
```

```
res <- kkmedias(dfcompleto, centers, dist, 5)
```

Finalmente:

```
#Graficamos los clusters
plot(df$x, df$y, col = (res$clusters))
# Ahora graficamos los centroides
points(x = res$centers, col = 4:8, pch = 19, cex = 3)
```




```
#Generamos la matriz de confusion
MatrixConfusion <- matrizconfusion(df$class,res$clusters)
MatrixConfusion
```

```
##       Cluster
## Clase     1     2     3
##     1 99659    95   246
##     2     5 99974    21
##     3  1302  7807 90891
```

good_luck.csv

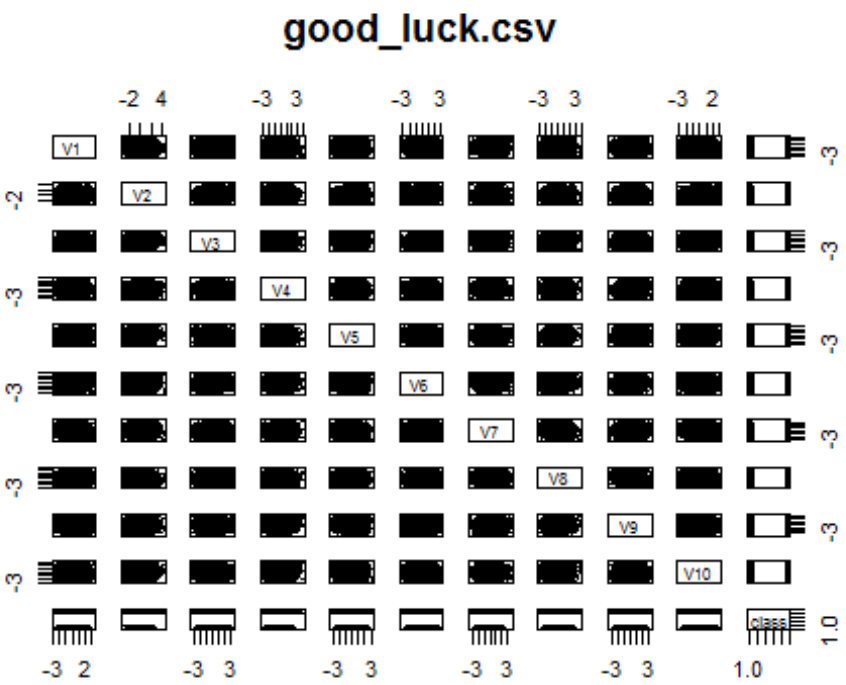
- Preprocesamiento:

```
setwd("C:/Users/Eric/Desktop/AprendizajeNoSupervisado")
library('FactoMineR')
name = "good_luck.csv"
#Lectura de datos.
df = read.csv(file =
```

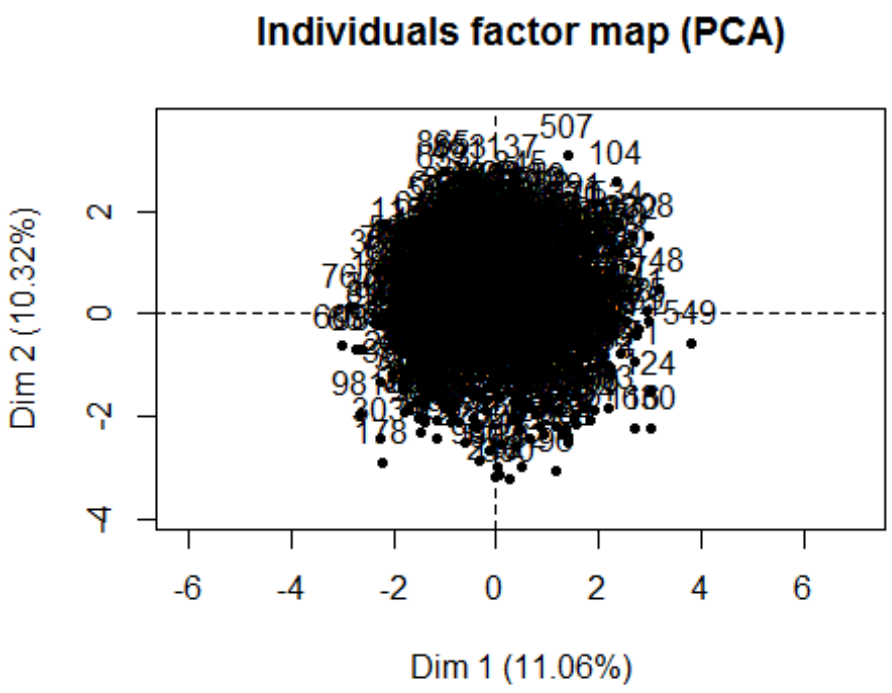
```
"C:/Users/Eric/Desktop/AprendizajeNoSupervisado/data/good_luck.csv", header =
F)
#Modificamos el nombre de las columnas por comodidad.
colnames(df)[11] <- "class"
#Coloco las clases del 1:n
df$class = as.numeric(df$class)
if (min(df$class) == 0){
  df$class <- df$class + 1
}
```

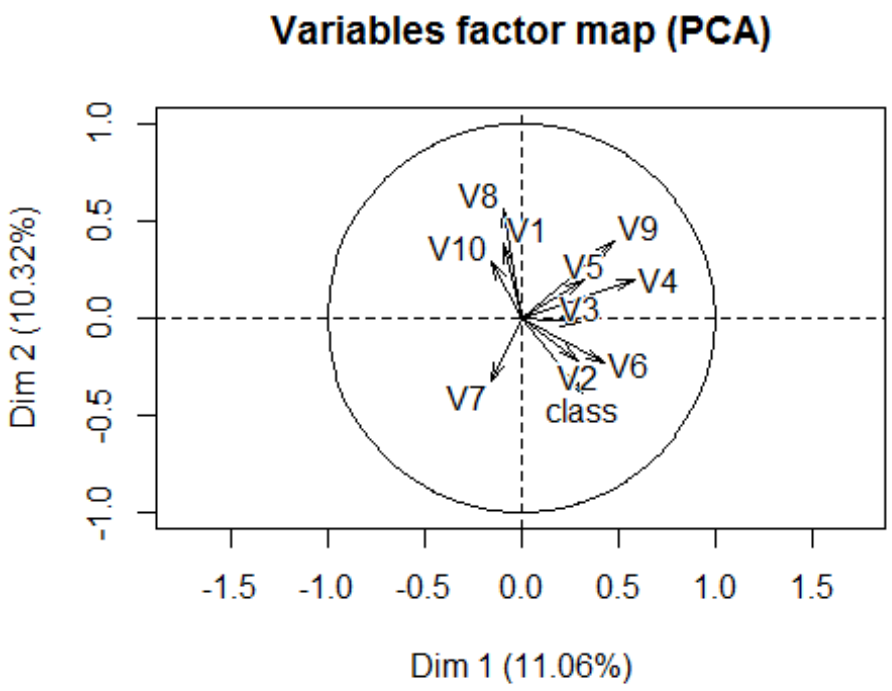
- Analisis exploratorio del dataset:

```
#**************************************************************************
************
#Analisis exploratorio del dataset
#**************************************************************************
************
#Podemos observar que hay 11 columnas.
head(df)

##          V1        V2        V3       V4        V5        V6        V7
## 1 -0.262989 -0.868793  0.133290 2.745286 -0.047937  1.357079 -0.499947
## 2  0.114041 -1.274678  1.518768 2.227560  0.679981 -0.257101  1.847423
## 3  0.974706 -0.314599 -0.731224 1.742879  0.361086  1.872469  0.874509
## 4  0.175839 -1.120091  0.522660 0.461655  1.264471  0.390200  0.714736
## 5 -0.238445 -0.656013 -0.663740 0.879592 -0.176941  0.915897 -0.327490
## 6 -0.162517  1.680123  1.188765 0.864624 -0.393824  1.630972  0.614608
##          V8        V9       V10 class
## 1 -1.874985 -0.395397  1.563203     2
## 2  0.586754  1.978936 -0.392775     2
## 3 -0.297938 -0.906587 -1.081991     2
## 4 -0.905856  0.456372 -0.697988     1
## 5  1.034972  0.732777 -0.257709     1
## 6  0.603824  0.294797 -0.649450     1

#Observamos cuantos elementos hay de cada clase.
table(df$class)

##
##   1   2
## 513 487

#1   2
#513 487

#Grafico
plot(df, main = name)
```

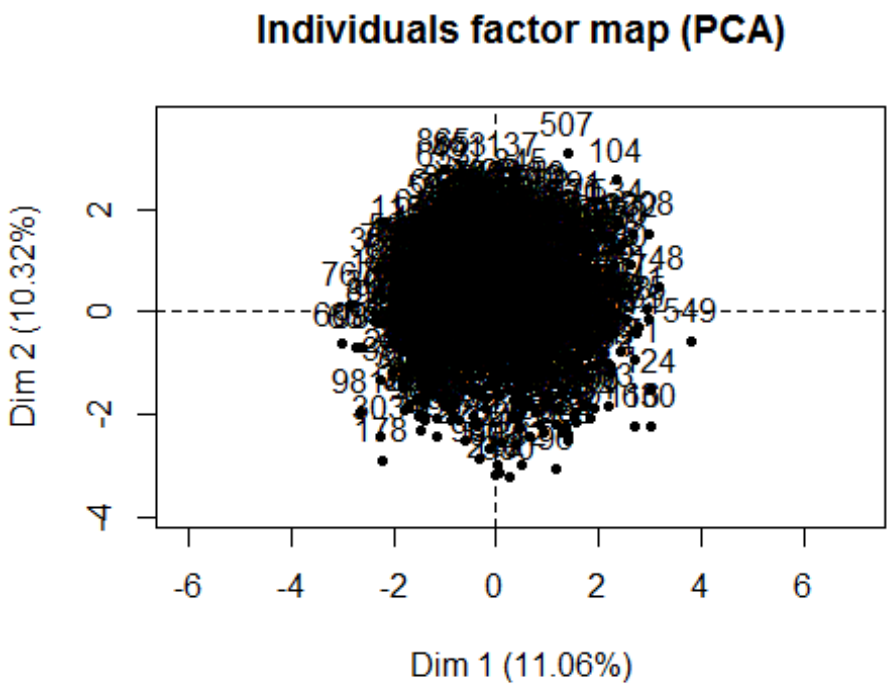

good_luck.csv
-2 4
-3 3
-3 3
-3 3
-3 2
V1
V2
V3
V4
V5
V6
V7
V8
V9
V10
class
-3 2
-3 3
-3 3
-3 3
-3 3
1.0

```
PCA <- PCA(df)
```


Individuals factor map (PCA)
Dim 2 (10.32%)
Dim 1 (11.06%)
507
104
549
24
-6
-4
-2
0
2
4
6
-4
-2
0
2


Variables factor map (PCA)
Dim 2 (10.32%)
Dim 1 (11.06%)
V8
V1
V9
V10
V5
V4
V3
V6
V2
V7
class
-1.5
-1.0
-0.5
0.0
0.5
1.0
1.5
-1.0
-0.5
0.0
0.5
1.0

```
plot(PCA)
```



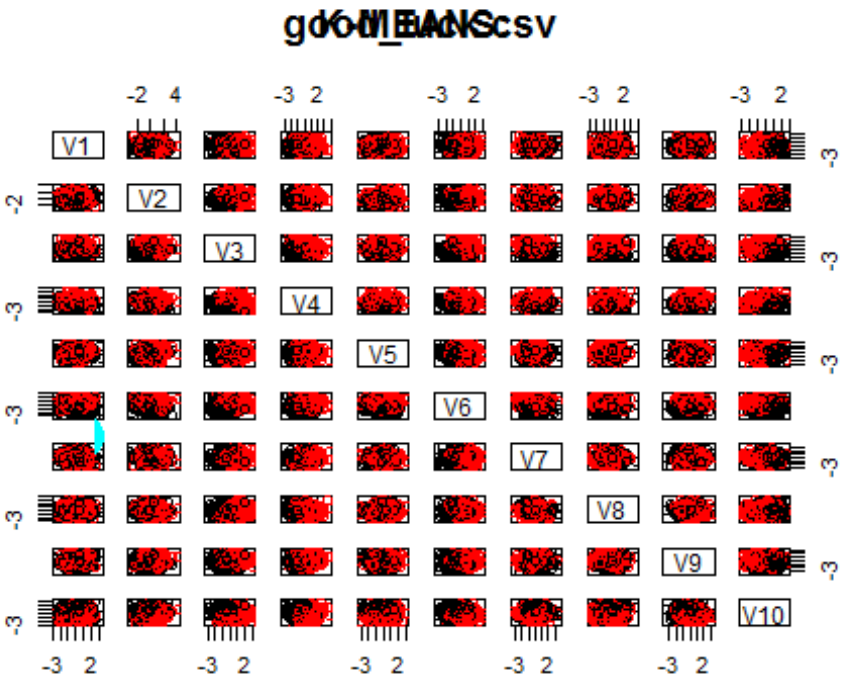

```
length(unique(df$class))
```

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

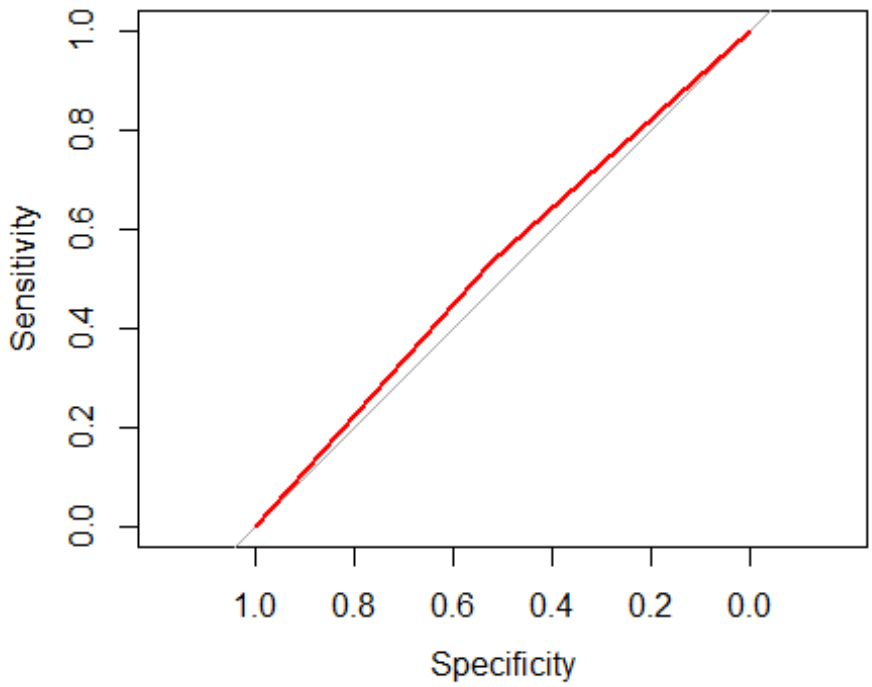
```
#Existen 2 clases.
```

K-medias:

```
#*************************************************************************
************
                                        #K-MEDIAS
#*************************************************************************
************
#Aplicamos k=2 ya que  existen 2 clases.
#kmedias(Dataframe, Columnas,K)
modeloK <- kmedias(df, 1:10, 2, name)
```




```
#Generamos la matriz de confusion
MatrixConfusionK <- matrizconfusion(df$class,modeloK$cluster)
MatrixConfusionK

##      Cluster
## Clase   1   2
##     1 270 243
##     2 230 257

#Calculamos la precision del modelo
PrecisionK <- precision(MatrixConfusionK)
PrecisionK

## [1] 0.527

#Generamos la curva de ROC
modeloKROC <- roc(df$class, modeloK$cluster)
plot(modeloKROC,type="l",col="red")
```



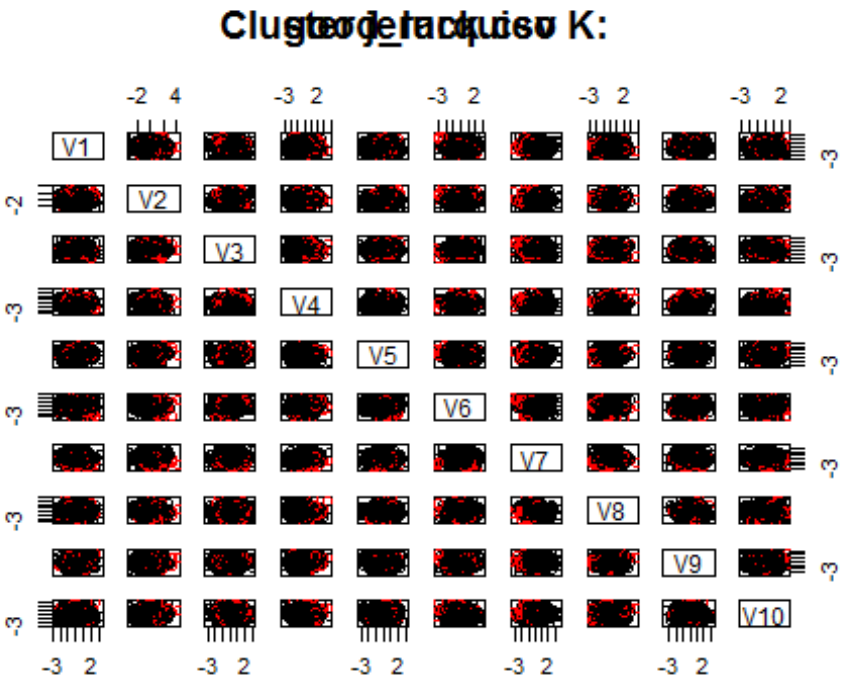

```
##
## Call:
## roc.default(response = df$class, predictor = modeloK$cluster)
##
## Data: modeloK$cluster in 513 controls (df$class 1) < 487 cases (df$class
2).
## Area under the curve: 0.527
```

Clusters jerárquicos:

```
#*************************************************************************
************
                                    #Cluster Jerarquicos
#*************************************************************************
************
#Copy del dataset
datos = df
#Elimino la columna clase para realizar aprendizaje no supervisado.
datos$class <- NULL
#Convierto el dataframe en una matriz
datos= as.matrix(datos)
#Calculamos la matriz de distancia
distancia = dist(datos)
```

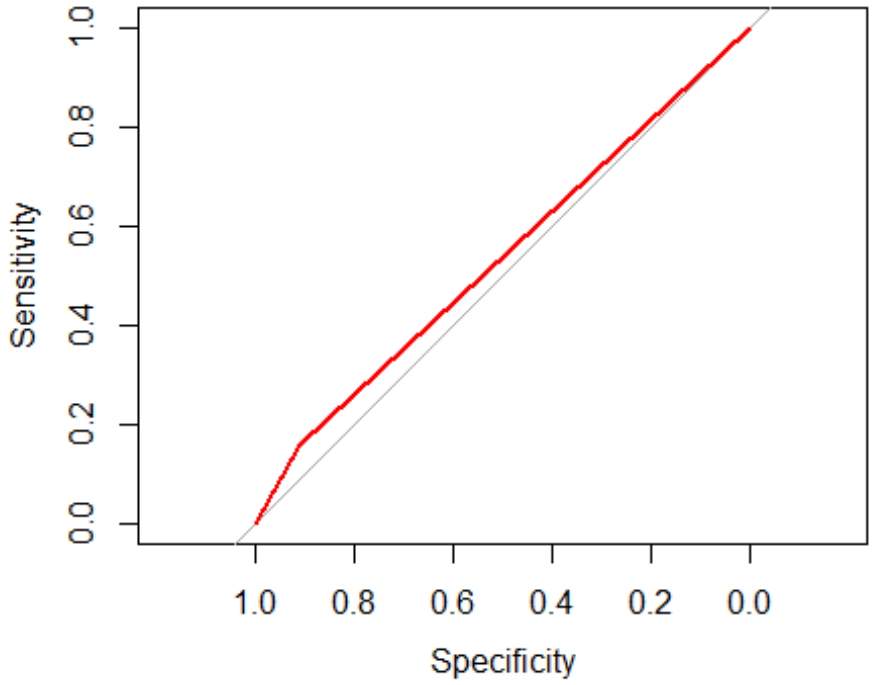
Método complete:

```
#-------------------------------------------------------------------------
----------
                                    #METHOD COMPLETE
#-------------------------------------------------------------------------
----------
#Dado un número de clústers k determinar la altura requerida
#para que tengamos el número de clúster k.
clustersD <- clusterJD(df, distancia, 1:10, "complete", 2, name)
```



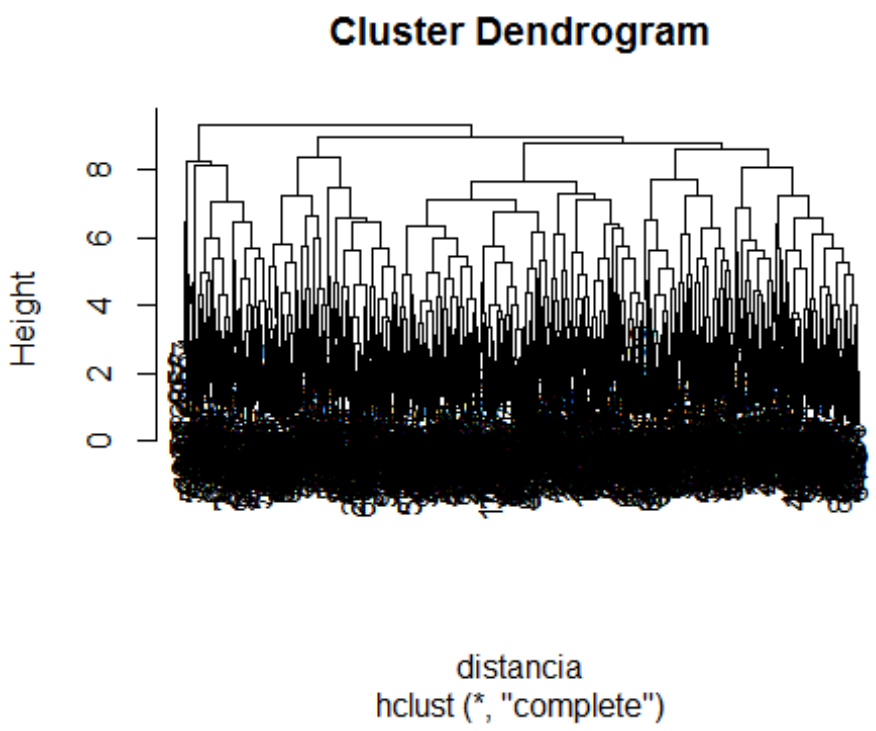

```
#Generamos la matriz de confusion
MatrixConfusionCJDC <- matrizconfusion(df$class, clustersD)
MatrixConfusionCJDC

##      Cluster
## Clase   1   2
##     1 467  46
##     2 409  78

#Calculamos la precision del modelo
PrecisionDC <- precision(MatrixConfusionCJDC)
PrecisionDC

## [1] 0.545
```

```r
#Generamos la curva de ROC
modeloDC <- roc(df$class, clustersD)
plot(modeloDC,type="l",col="red")
```

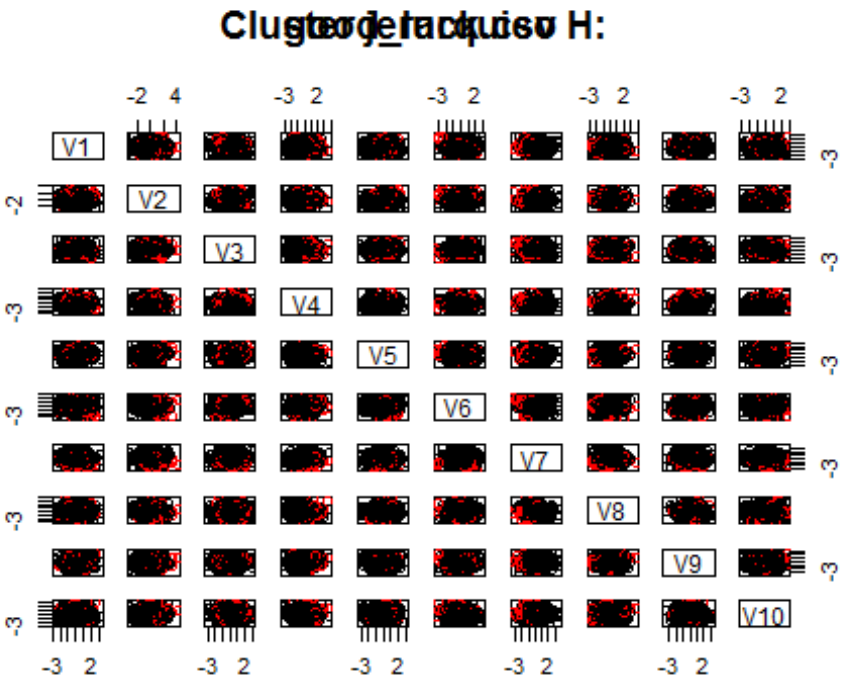



```
##
## Call:
## roc.default(response = df$class, predictor = clustersD)
##
## Data: clustersD in 513 controls (df$class 1) < 487 cases (df$class 2).
## Area under the curve: 0.5352
```

```r
#*********************************************************
#Dada una altura h (una medida de disimilaridad) determinar
#el número de clústers que se obtienen.
clustersH <- clusterJH(df, distancia, 1:10, "complete", 9.3, name)
```


Cluster Dendrogram
Height
0
2
4
6
8
distancia
hclust (*, "complete")

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


V1
V2
V3
V4
V5
V6
V7
V8
V9
V10

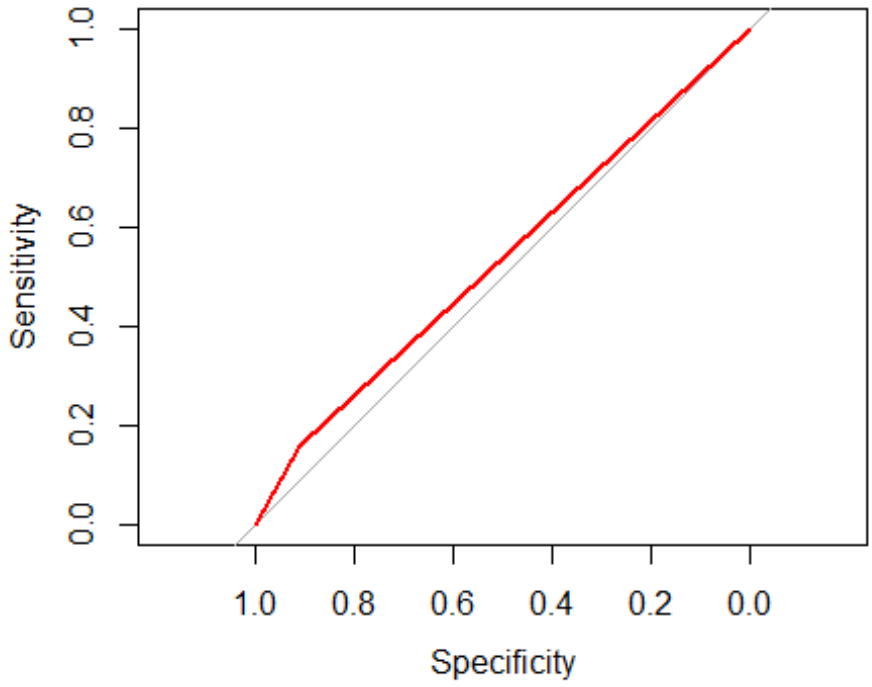
```r
#Generamos la matriz de confusion
MatrixConfusionCJHC <- matrizconfusion(df$class, clustersH)
MatrixConfusionCJHC
```

```
##      Cluster
## Clase   1   2
##     1 467  46
##     2 409  78
```

```r
#Calculamos la precision del modelo
PrecisionHC <- precision(MatrixConfusionCJHC)
PrecisionHC
```

```
## [1] 0.545
```

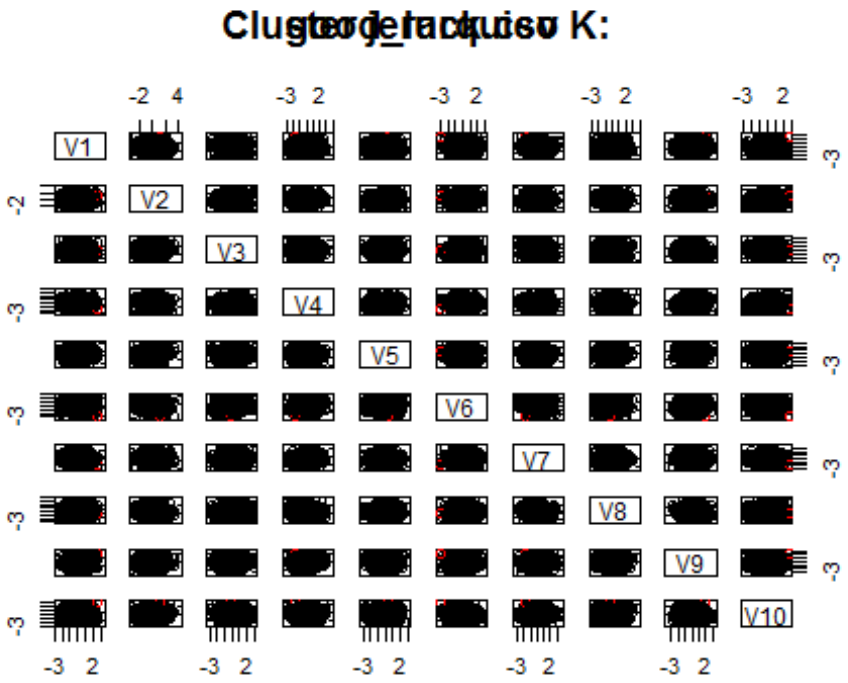
```r
#Generamos la curva de ROC
modeloHC <- roc(df$class, clustersH)
plot(modeloHC,type="l",col="red")
```




```
##
## Call:
## roc.default(response = df$class, predictor = clustersH)
##
## Data: clustersH in 513 controls (df$class 1) < 487 cases (df$class 2).
## Area under the curve: 0.5352
```

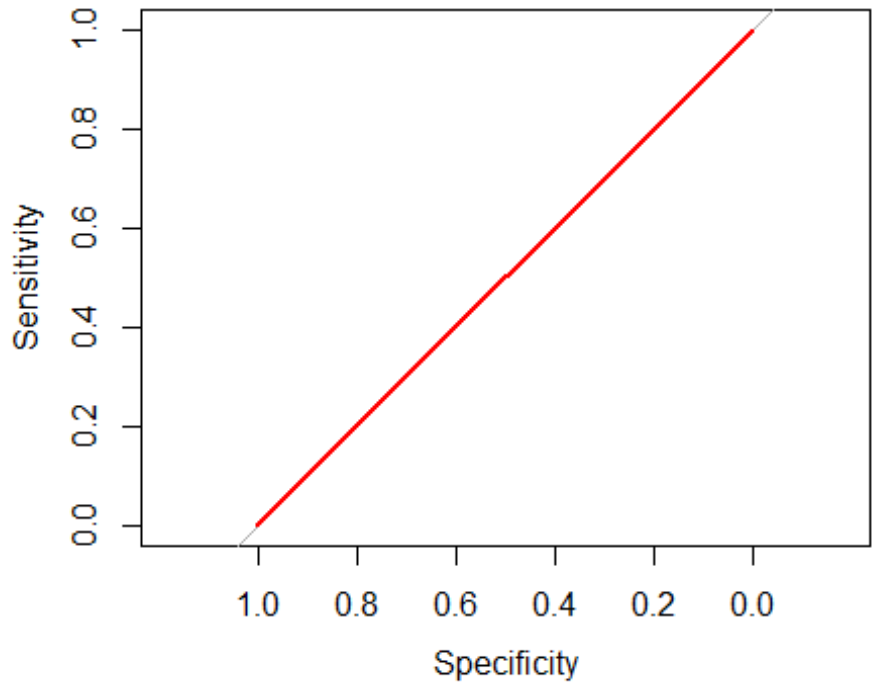
Método single:

```
#-------------------------------------------------------------------------
----------
                                    #METHOD SINGLE
#-------------------------------------------------------------------------
----------
#Dado un número de clústers k determinar la altura requerida
#para que tengamos el número de clúster k.
clustersD <- clusterJD(df, distancia, 1:10, "single", 2, name)
```



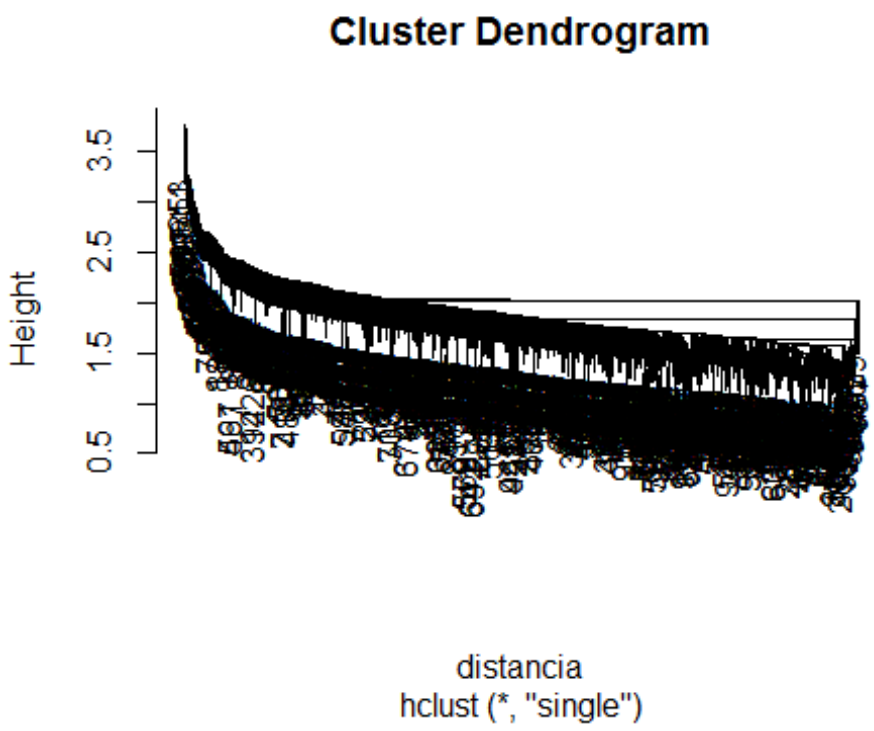

```
#Generamos la matriz de confusion
MatrixConfusionCJDS <- matrizconfusion(df$class, clustersD)
MatrixConfusionCJDS

##      Cluster
## Clase   1   2
##     1 513   0
##     2 486   1

#Calculamos la precision del modelo
PrecisionDS <- precision(MatrixConfusionCJDS)
PrecisionDS

## [1] 0.514
```

```r
#Generamos la curva de ROC
modeloDS <- roc(df$class, clustersD)
plot(modeloDS,type="l",col="red")
```

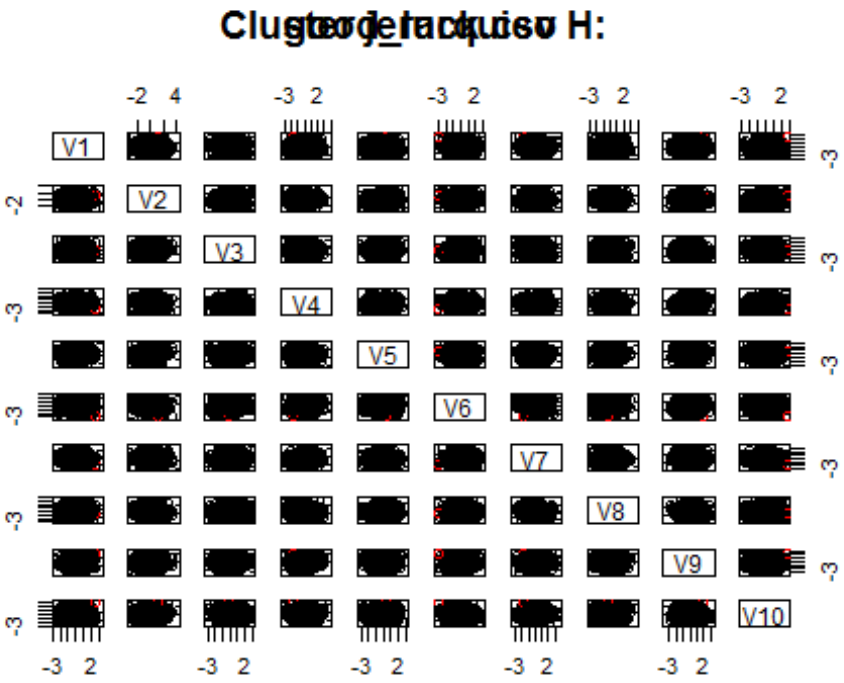



```
##
## Call:
## roc.default(response = df$class, predictor = clustersD)
##
## Data: clustersD in 513 controls (df$class 1) < 487 cases (df$class 2).
## Area under the curve: 0.501
```

```r
#*********************************************************
#Dada una altura h (una medida de disimilaridad) determinar
#el número de clústers que se obtienen.
clustersH <- clusterJH(df, distancia, 1:10, "single", 3.75, name)
```


Cluster Dendrogram
Height
0.5
1.5
2.5
3.5
distancia
hclust (*, "single")

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


V1
V2
V3
V4
V5
V6
V7
V8
V9
V10

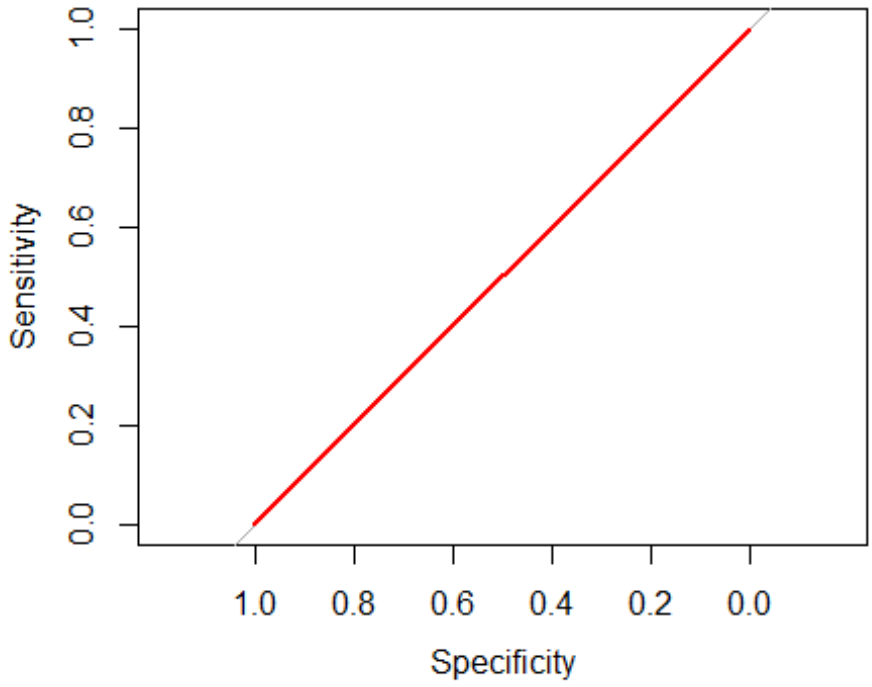
```r
#Generamos la matriz de confusion
MatrixConfusionCJHS <- matrizconfusion(df$class, clustersH)
MatrixConfusionCJHS
```

```
##      Cluster
## Clase   1   2
##     1 513   0
##     2 486   1
```

```r
#Calculamos la precision del modelo
PrecisionHS <- precision(MatrixConfusionCJHS)
PrecisionHS
```

```
## [1] 0.514
```

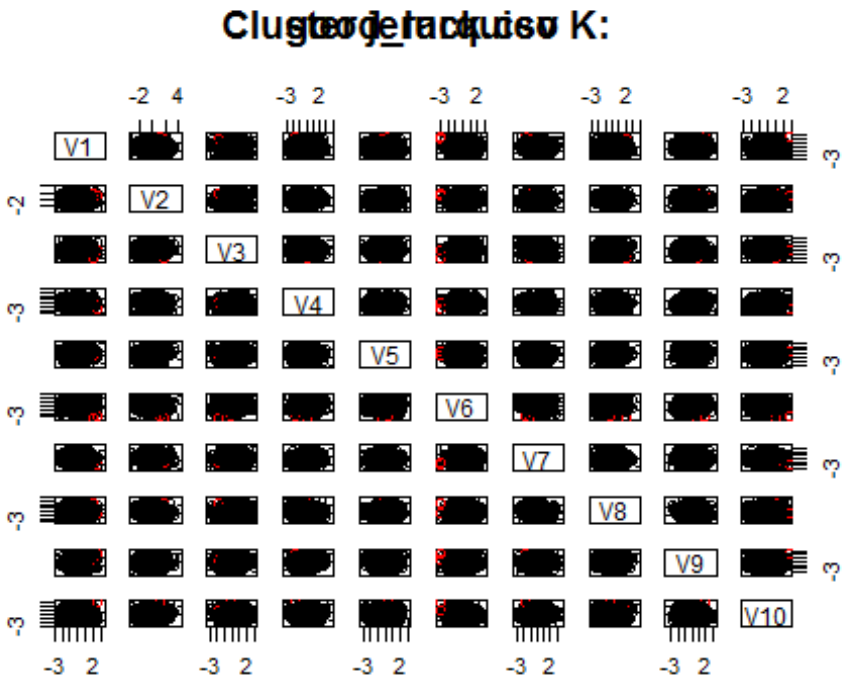
```r
#Generamos la curva de ROC
modeloHS <- roc(df$class, clustersH)
plot(modeloHS,type="l",col="red")
```




```
##
## Call:
## roc.default(response = df$class, predictor = clustersH)
##
## Data: clustersH in 513 controls (df$class 1) < 487 cases (df$class 2).
## Area under the curve: 0.501
```

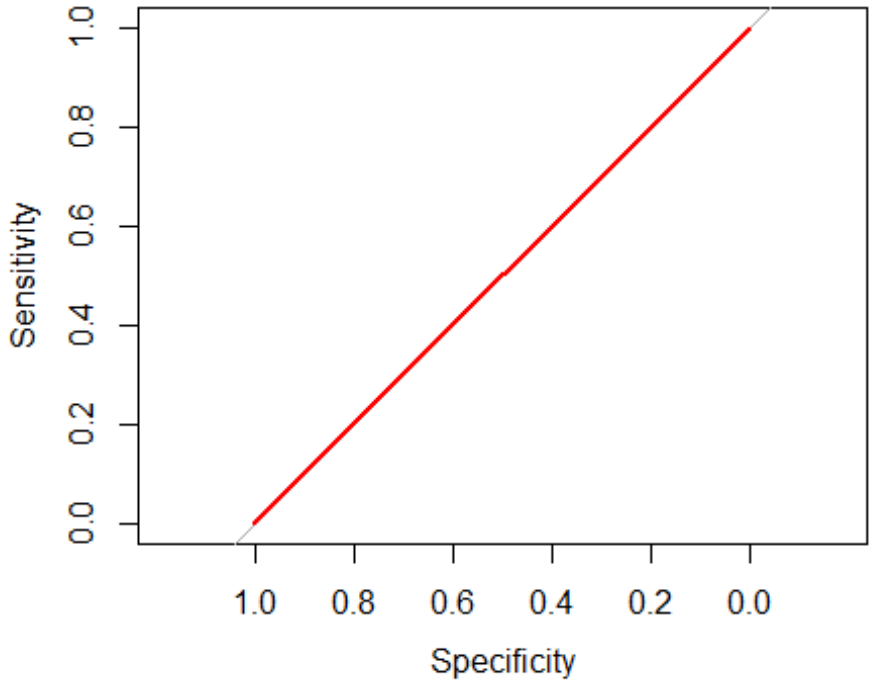
Método average:

```
#-------------------------------------------------------------------------------
----------
                                    #METHOD AVERAGE
#-------------------------------------------------------------------------------
----------
#Dado un número de clústers k determinar la altura requerida
#para que tengamos el número de clúster k.
clustersD <- clusterJD(df, distancia, 1:10, "average", 2, name)
```



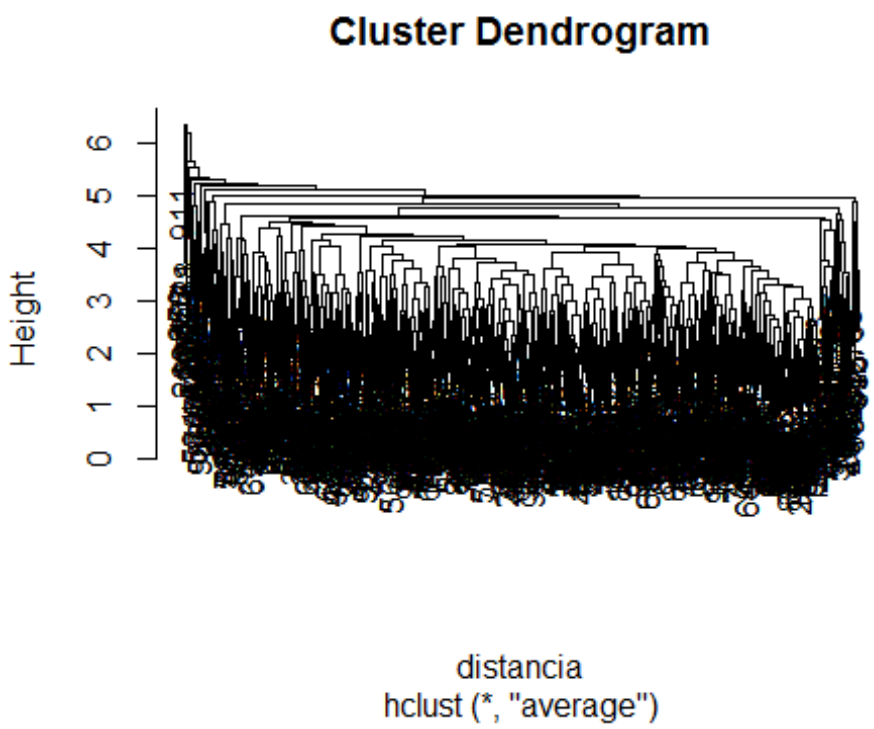

```
#Generamos la matriz de confusion
MatrixConfusionCJDA <- matrizconfusion(df$class, clustersD)
MatrixConfusionCJDA

##      Cluster
## Clase   1   2
##     1 513   0
##     2 485   2

#Calculamos la precision del modelo
PrecisionDA <- precision(MatrixConfusionCJDA)
PrecisionDA

## [1] 0.515
```

```r
#Generamos la curva de ROC
modeloDA <- roc(df$class, clustersD)
plot(modeloDA,type="l",col="red")
```

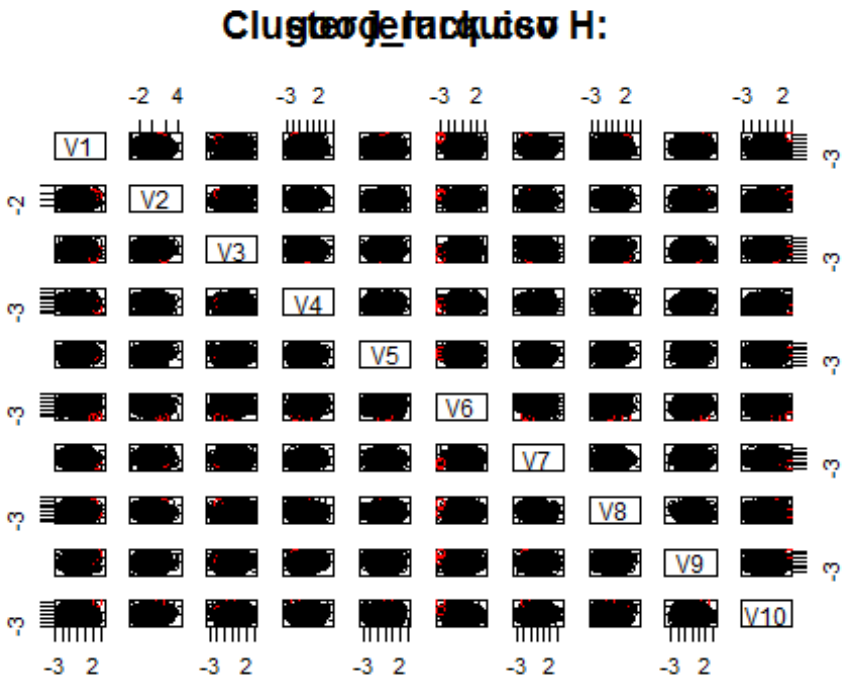



```
##
## Call:
## roc.default(response = df$class, predictor = clustersD)
##
## Data: clustersD in 513 controls (df$class 1) < 487 cases (df$class 2).
## Area under the curve: 0.5021
```

```r
#*********************************************************
#Dada una altura h (una medida de disimilaridad) determinar
#el número de clústers que se obtienen.
clustersH <- clusterJH(df, distancia, 1:10, "average", 6.2, name)
```


Cluster Dendrogram
Height
0
1
2
3
4
5
6
distancia
hclust (*, "average")

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


V1
V2
V3
V4
V5
V6
V7
V8
V9
V10

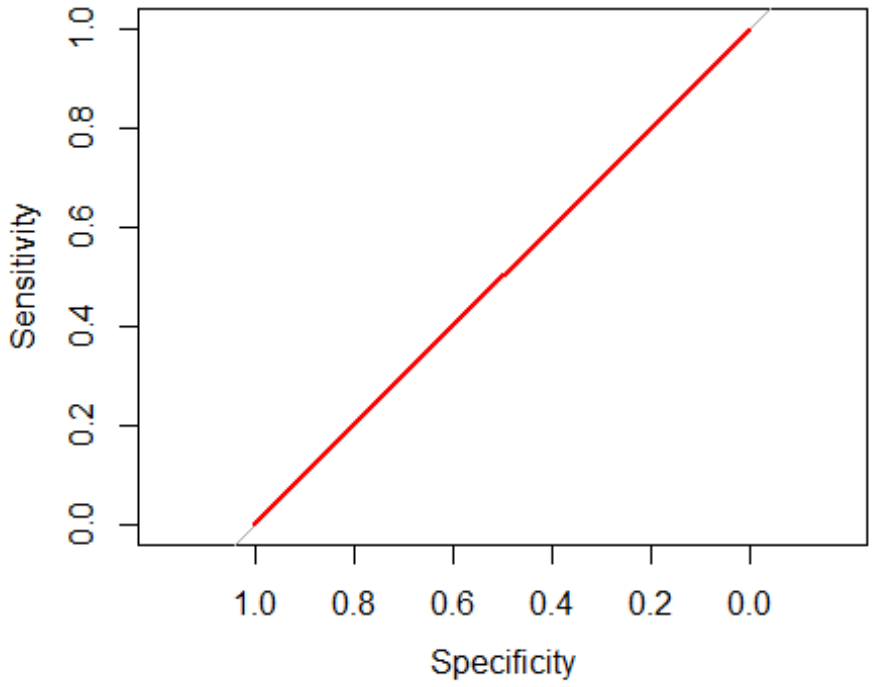
```r
#Generamos la matriz de confusion
MatrixConfusionCJHA <- matrizconfusion(df$class, clustersH)
MatrixConfusionCJHA
```

```
##      Cluster
## Clase   1   2
##     1 513   0
##     2 485   2
```

```r
#Calculamos la precision del modelo
PrecisionHA <- precision(MatrixConfusionCJHA)
PrecisionHA
```

```
## [1] 0.515
```

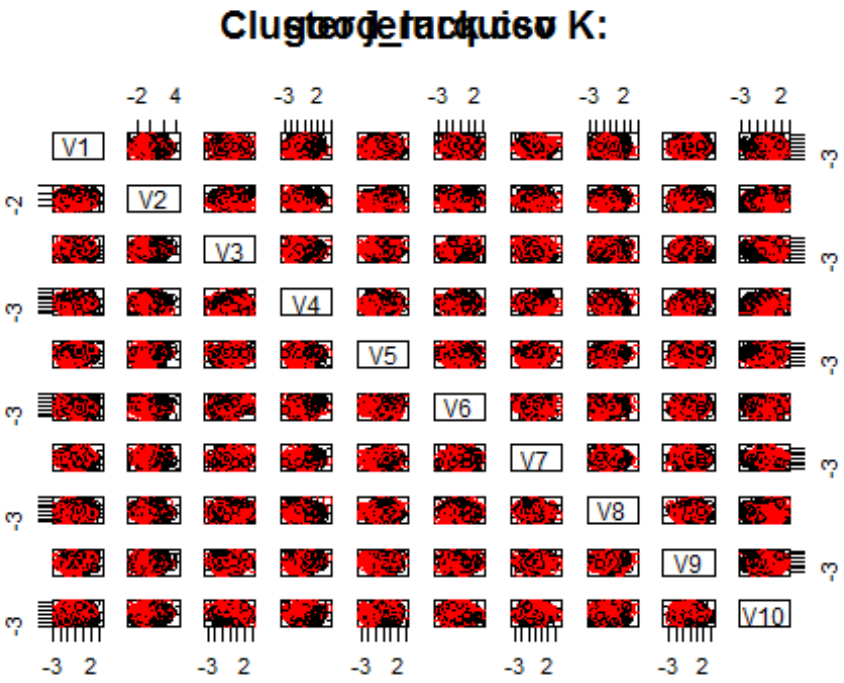
```r
#Generamos la curva de ROC
modeloHA <- roc(df$class, clustersH)
plot(modeloHA,type="l",col="red")
```




```
##
## Call:
## roc.default(response = df$class, predictor = clustersH)
##
## Data: clustersH in 513 controls (df$class 1) < 487 cases (df$class 2).
## Area under the curve: 0.5021
```

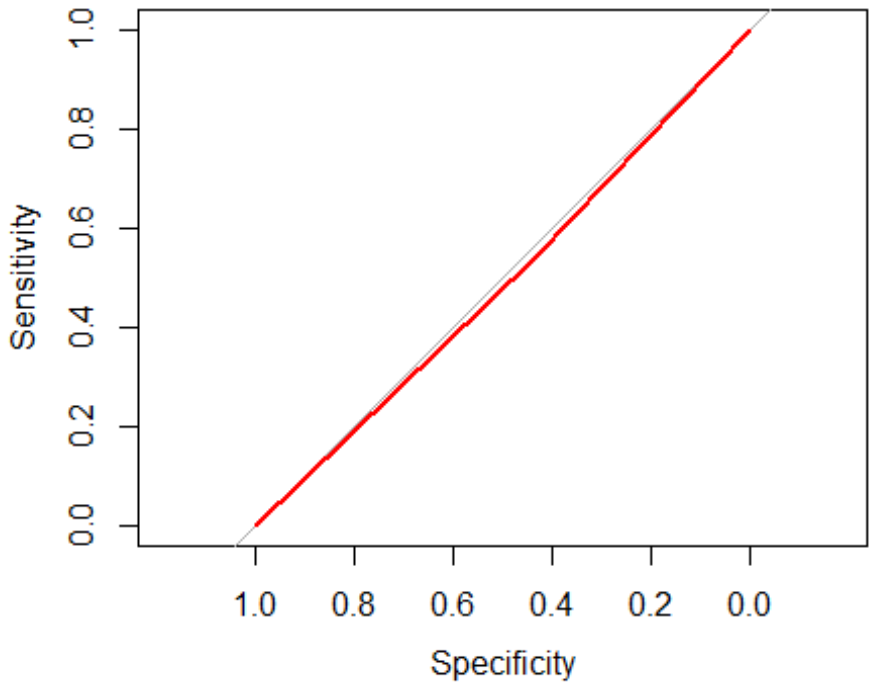
Método ward.D

```
#-------------------------------------------------------------------------------
----------
                                        #METHOD ward.D
#-------------------------------------------------------------------------------
----------
#Dado un número de clústers k determinar la altura requerida
#para que tengamos el número de clúster k.
clustersD <- clusterJD(df, distancia, 1:10, "ward.D", 2, name)
```



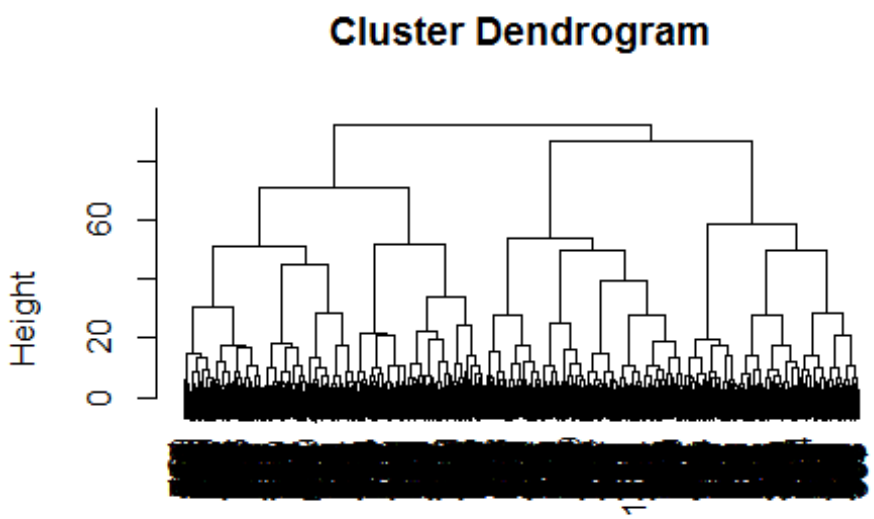

```
#Generamos la matriz de confusion
MatrixConfusionCJDW <- matrizconfusion(df$class, clustersD)
MatrixConfusionCJDW

##      Cluster
## Clase   1   2
##     1 290 223
##     2 264 223

#Calculamos la precision del modelo
PrecisionDW <- precision(MatrixConfusionCJDW)
PrecisionDW

## [1] 0.513
```

```r
#Generamos la curva de ROC
modeloDW <- roc(df$class, clustersD)
plot(modeloDW,type="l",col="red")
```

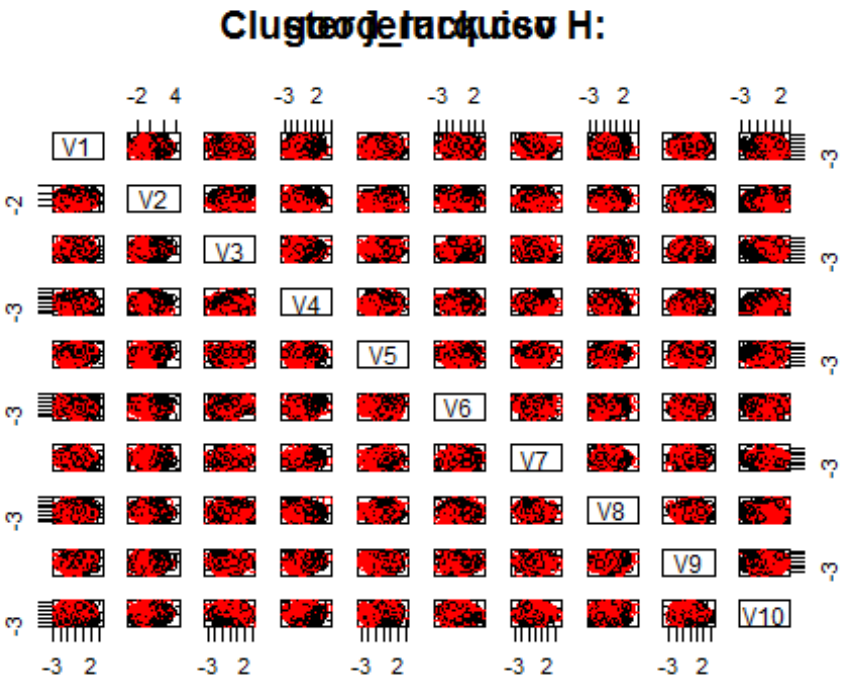



```
##
## Call:
## roc.default(response = df$class, predictor = clustersD)
##
## Data: clustersD in 513 controls (df$class 1) < 487 cases (df$class 2).
## Area under the curve: 0.4884
```

```r
#*********************************************************
#Dada una altura h (una medida de disimilaridad) determinar
#el número de clústers que se obtienen.
clustersH <- clusterJH(df, distancia, 1:10, "ward.D", 90, name)
```


Cluster Dendrogram
Height
0
20
60
distancia
hclust (*, "ward.D")

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


V1
V2
V3
V4
V5
V6
V7
V8
V9
V10

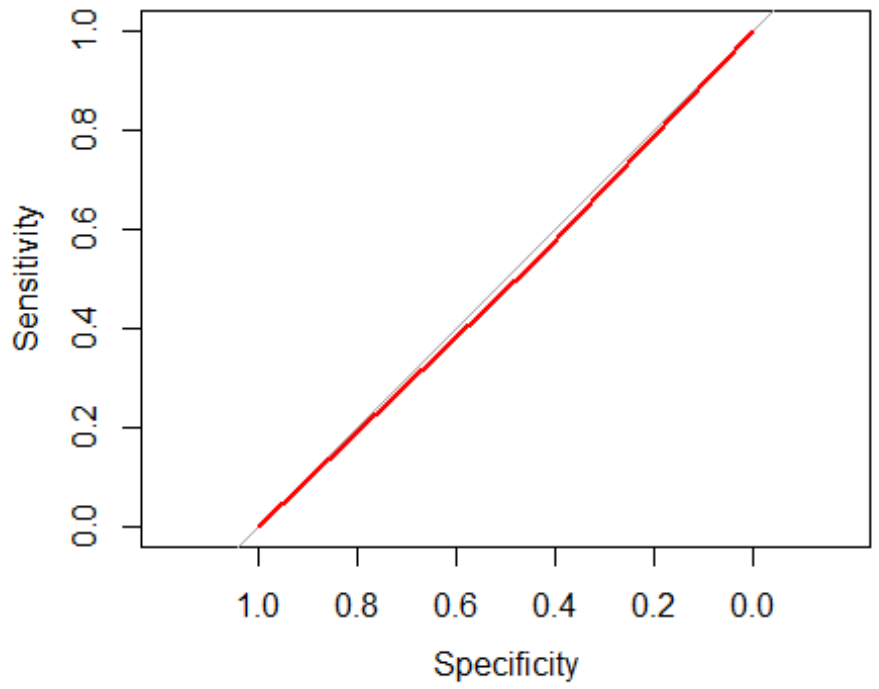
```r
#Generamos la matriz de confusion
MatrixConfusionCJHW <- matrizconfusion(df$class, clustersH)
MatrixConfusionCJHW
```

```
##      Cluster
## Clase   1   2
##     1 290 223
##     2 264 223
```

```r
#Calculamos la precision del modelo
PrecisionHW <- precision(MatrixConfusionCJHW)
PrecisionHW
```

```
## [1] 0.513
```

```r
#Generamos la curva de ROC
modeloHW <- roc(df$class, clustersH)
plot(modeloHW,type="l",col="red")
```




```
##
## Call:
## roc.default(response = df$class, predictor = clustersH)
##
## Data: clustersH in 513 controls (df$class 1) < 487 cases (df$class 2).
## Area under the curve: 0.4884
```

Mejor modelo

```
#-------------------------------------------------------------------------
----------
#                                  MEJOR MODELO
#-------------------------------------------------------------------------
----------
precisiones <- c(PrecisionK, PrecisionDC, PrecisionHC, PrecisionDS,
PrecisionHS, PrecisionDA,
                 PrecisionHA, PrecisionDW, PrecisionHW)
x <- which.max(precisiones)
mejormodelo <- bestmodel(x)
cat("El mejor modelo es: ", mejormodelo, ", que posee una precision de: ",
precisiones[x], ".")
```

```
## El mejor modelo es:  CLASIFICACION JERARQUICA K: METHOD COMPLETE , que
posee una precision de:  0.545 .
```

guess.csv

- Preprocesamiento:

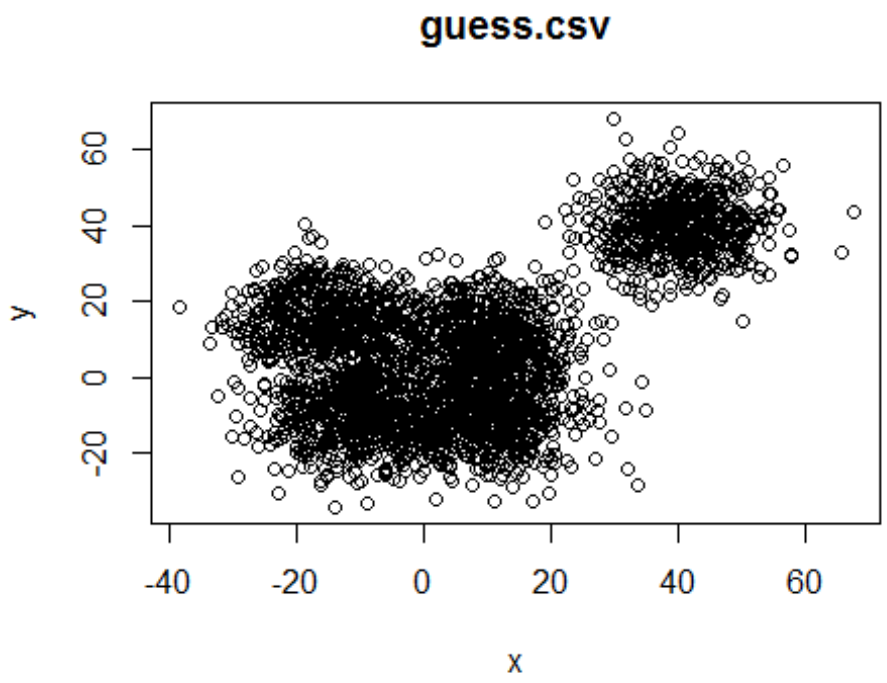
```
setwd("C:/Users/Eric/Desktop/AprendizajeNoSupervisado")
name = "guess.csv"
#Lectura de datos.
df = read.csv(file =
"C:/Users/Eric/Desktop/AprendizajeNoSupervisado/data/guess.csv", header = F)
#Modificamos el nombre de las columnas por comodidad.
colnames(df) <- c("x","y")
```

- Analisis exploratorio del dataset:

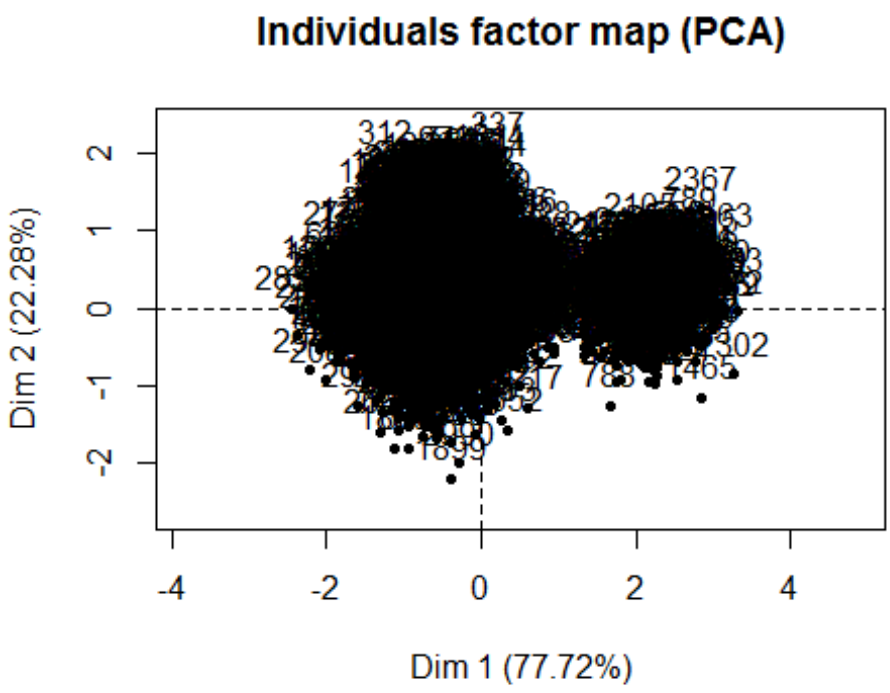
```
#*************************************************************************
************
#                           Analisis exploratorio del dataset
#*************************************************************************
************
#Podemos observar que hay 2 columnas.
head(df)
```

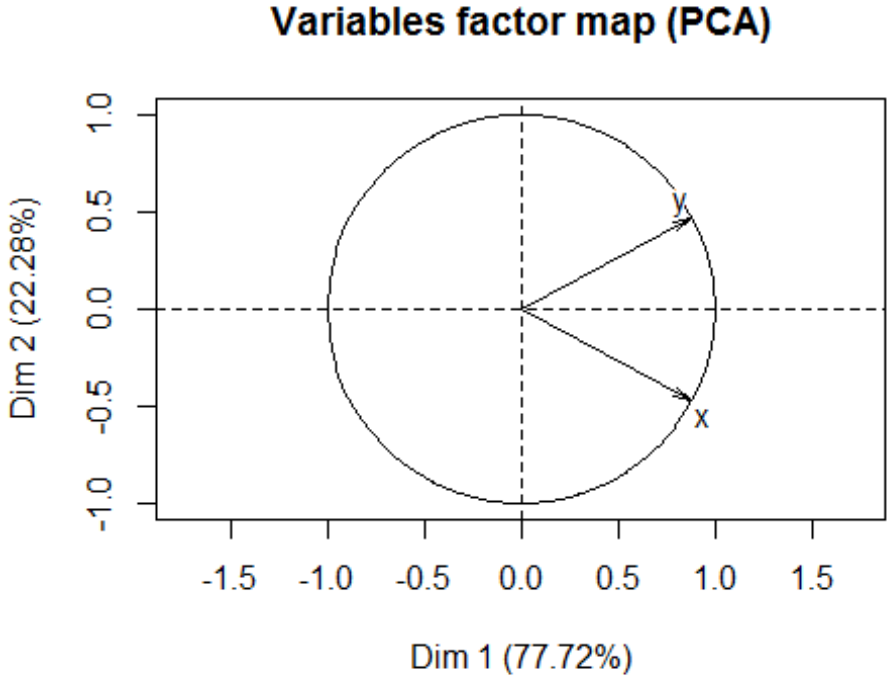
```
##            x          y
## 1 -22.856666 -30.302967
## 2  17.637588   8.613766
## 3 -24.046328  19.647426
## 4 -25.918317  12.200390
## 5   5.908121 -18.359366
## 6   8.395719 -15.811708
```

```
#Grafico
plot(df$x, df$y, xlab = "x", ylab = "y", main = name)
```

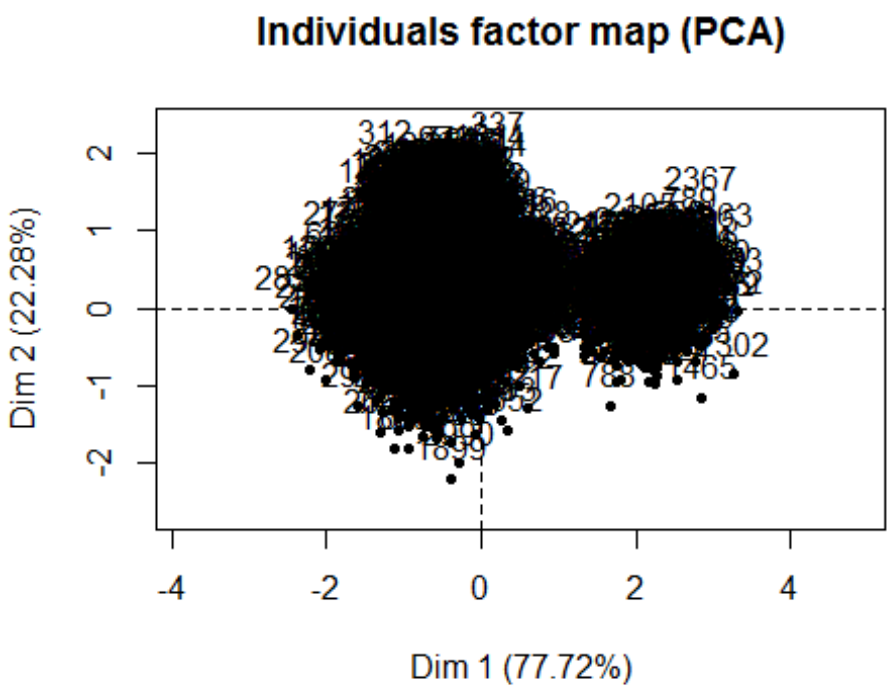



```
PCA <- PCA(df)
```

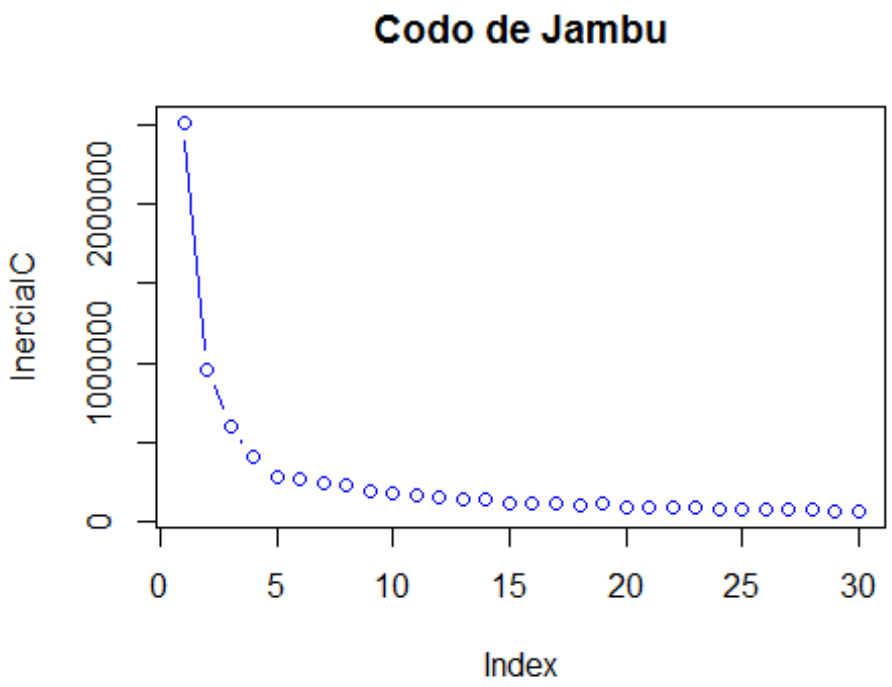

Individuals factor map (PCA)
Dim 2 (22.28%)
Dim 1 (77.72%)
-4
-2
0
2
4
-2
-1
0
1
2
2367
302


Variables factor map (PCA)
Dim 2 (22.28%)
Dim 1 (77.72%)
-1.5
-1.0
-0.5
0.0
0.5
1.0
1.5
-1.0
-0.5
0.0
0.5
1.0
y
x

```
plot(PCA)
```



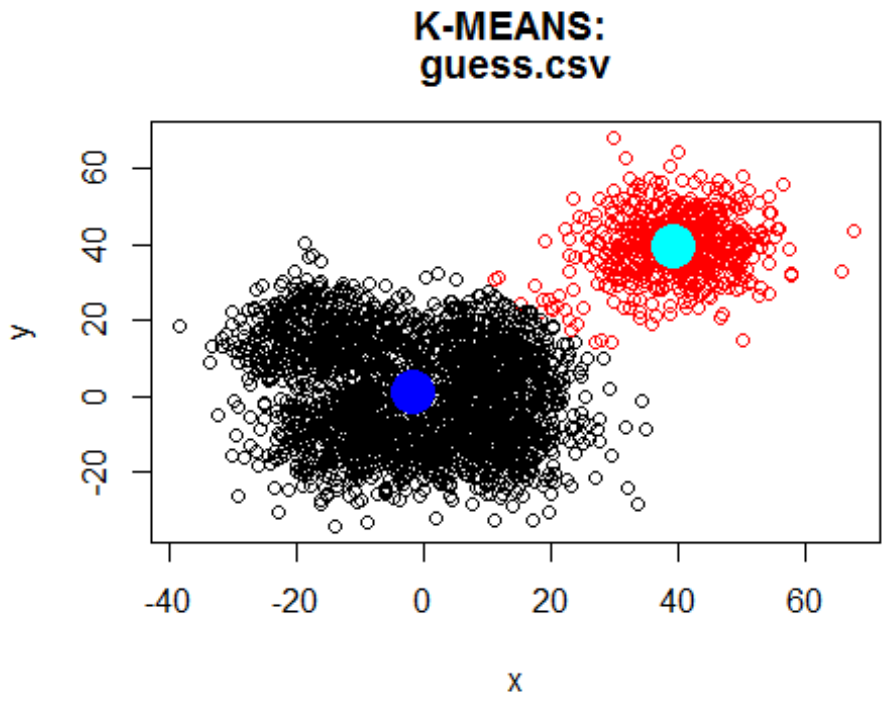

```
#Podemos observar 2 conglomerados

#Aplicamos codo de Jambu como ayuda para seleccionar el K
InerciaIC = rep(0, 30)
for (k in 1:30) {
  grupos = kmeans(df, k)
  InerciaIC[k] = grupos$tot.withinss
}

## Warning: did not converge in 10 iterations

plot(InerciaIC, col = "blue", type = "b", main = "Codo de Jambu")
```




```
#Segun mi analisis el k adecuado es 2
```

K-medias:

```
#*************************************************************************
************
                                                #K-MEDIAS
#*************************************************************************
************
#Aplicamos k=2 ya que identificamos 2 conglomerados.
#kmedias(Dataframe, Columnas,K)
modeloK <- kmedias(df, 1:2, 2, name)
```

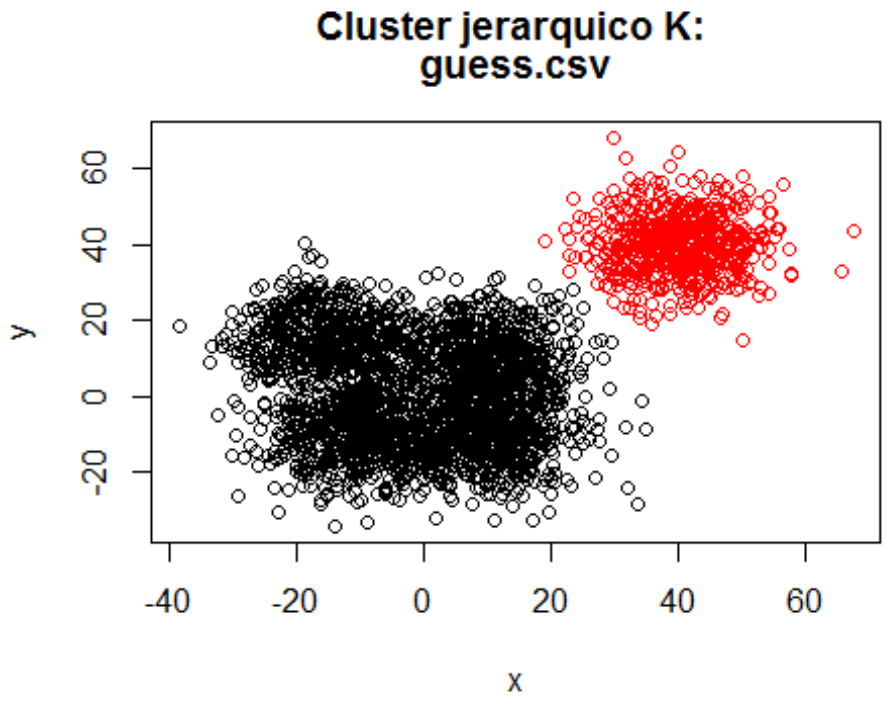


Clusters jerárquicos:

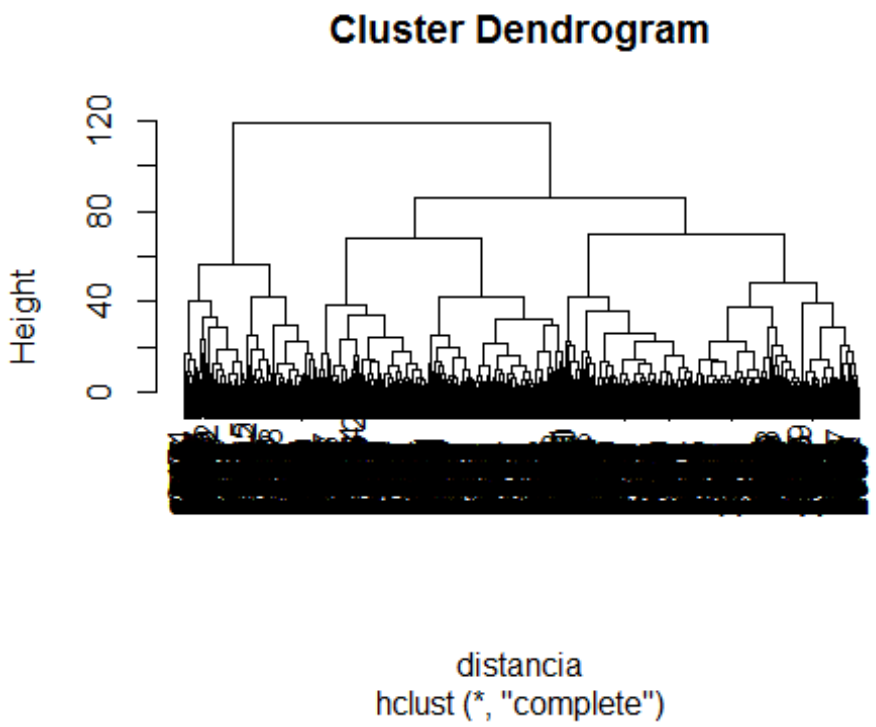
```
#*************************************************************************
************
                                    #CLUSTER JERARQUICO
#*************************************************************************
************
#Copy del dataset
datos = df
#Elimino la columna clase para realizar aprendizaje no supervisado.
datos$class <- NULL
#Convierto el dataframe en una matriz
datos= as.matrix(datos)
#Calculamos la matriz de distancia
distancia = dist(datos)
```

Método complete:

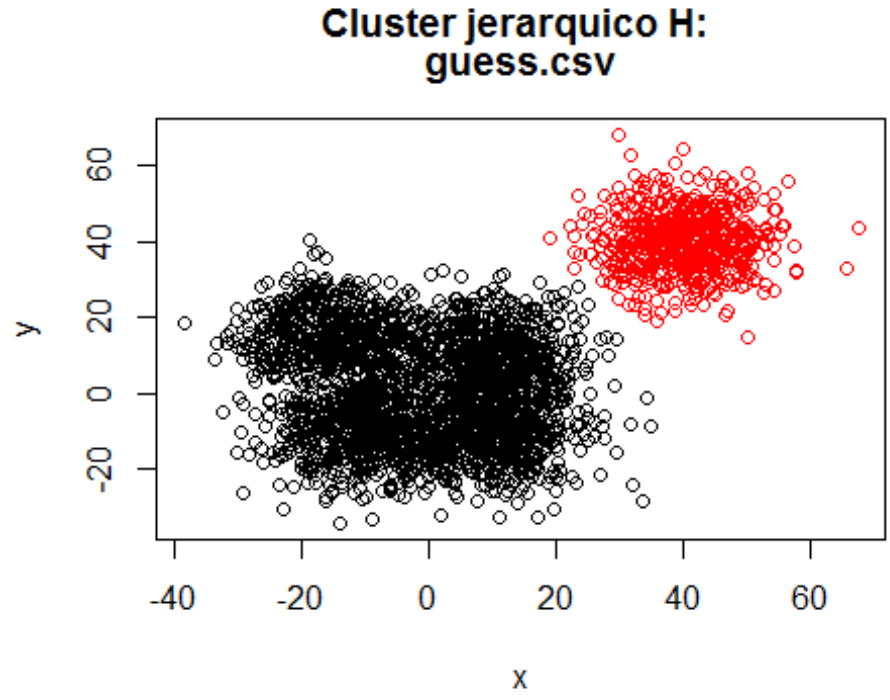
```
#-------------------------------------------------------------------------
----------
                                    #METHOD COMPLETE
#-------------------------------------------------------------------------
----------
#Dado un número de clústers k determinar la altura requerida
#para que tengamos el número de clúster k.
clustersD <- clusterJD(df, distancia, 1:2, "complete", 2, name)
```




```r
#*********************************************************
#Dada una altura h (una medida de disimilaridad) determinar
#el número de clústers que se obtienen.
clustersH <- clusterJH(df, distancia, 1:2, "complete", 100, name)
```

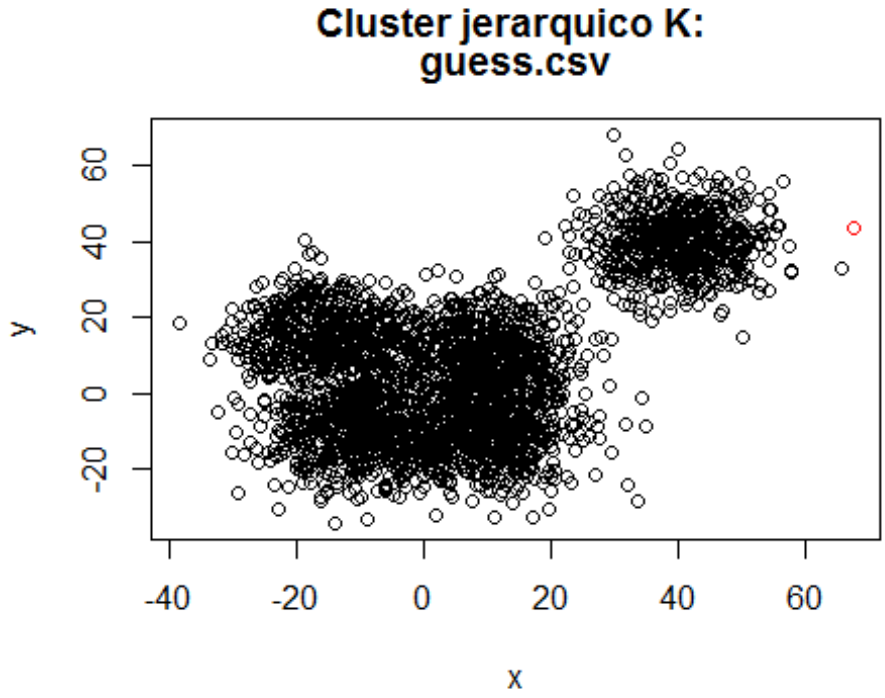

Cluster Dendrogram
Height
0
40
80
120
distancia
hclust (*, "complete")

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

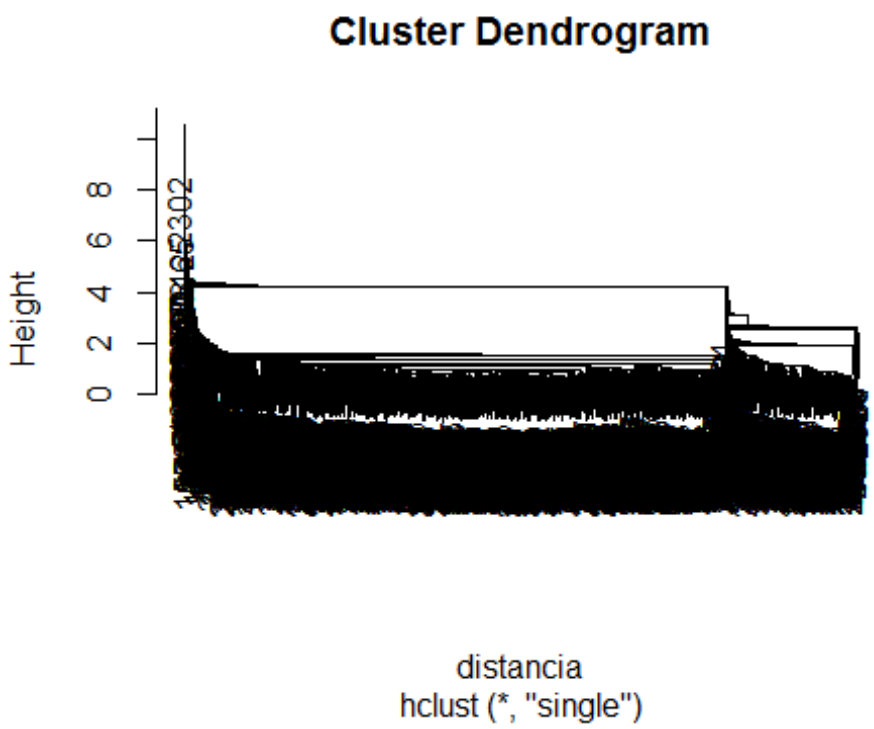

Cluster jerarquico H:
guess.csv
y
-20
0
20
40
60
-40
-20
0
20
40
60
x

Método single:

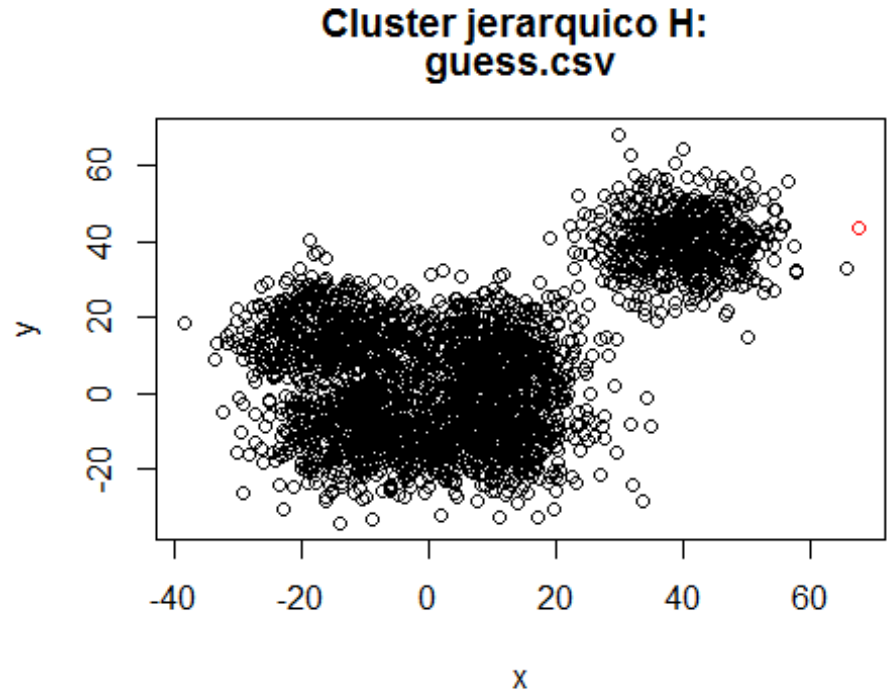
```
#-------------------------------------------------------------------------
----------
                                    #METHOD SINGLE
#-------------------------------------------------------------------------
----------
#Dado un número de clústers k determinar la altura requerida
#para que tengamos el número de clúster k.
clustersD <- clusterJD(df, distancia, 1:2, "single", 2, name)
```




```
#*********************************************************
#Dada una altura h (una medida de disimilaridad) determinar
#el número de clústers que se obtienen.
clustersH <- clusterJH(df, distancia, 1:2, "single", 10, name)
```

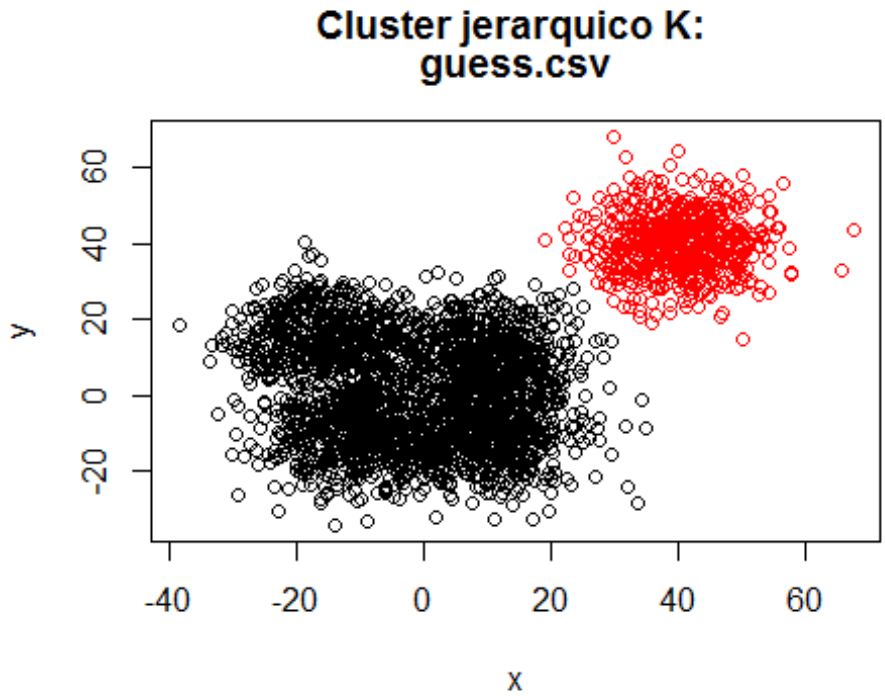

Cluster Dendrogram
Height
0
2
4
6
8
distancia
hclust (*, "single")

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

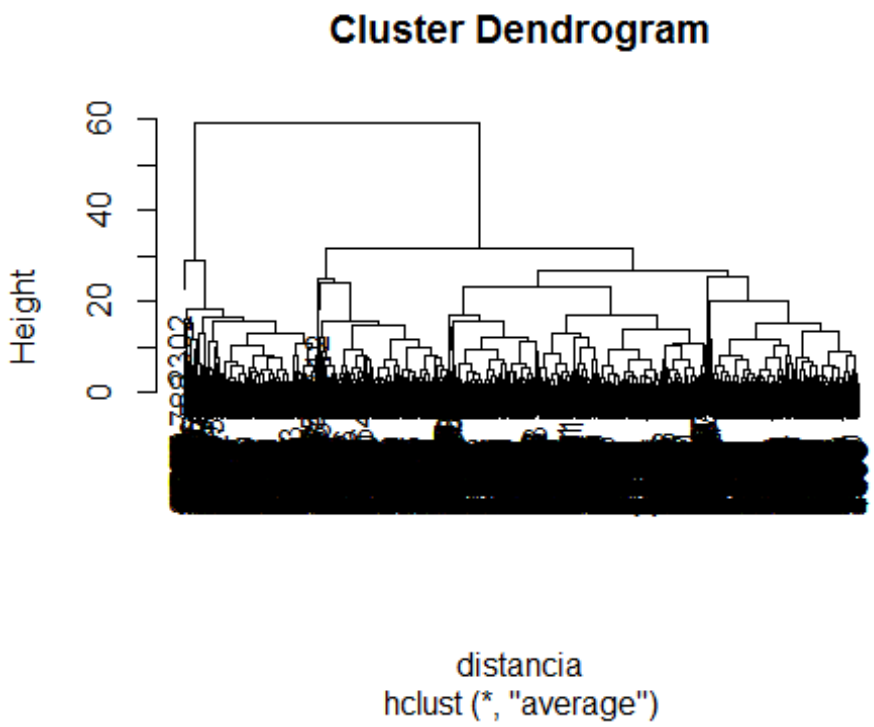

Cluster jerarquico H:
guess.csv
y
-20
0
20
40
60
-40
-20
0
20
40
60
x

Método average:

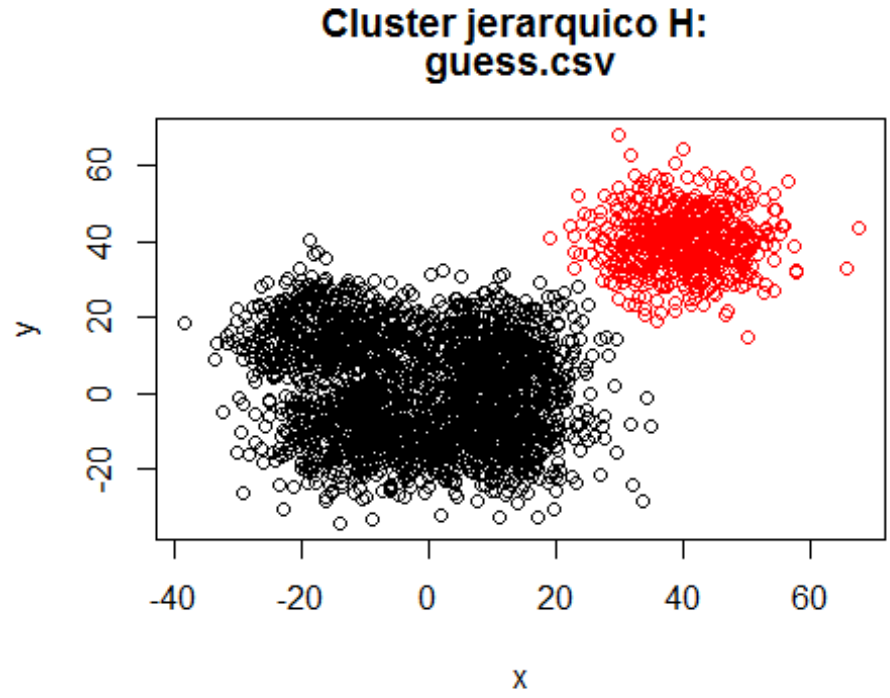
```
#-------------------------------------------------------------------------
----------
                                    #METHOD AVERAGE
#-------------------------------------------------------------------------
----------
#Dado un número de clústers k determinar la altura requerida
#para que tengamos el número de clúster k.
clustersD <- clusterJD(df, distancia, 1:2, "average", 2, name)
```




```
#*********************************************************
#Dada una altura h (una medida de disimilaridad) determinar
#el número de clústers que se obtienen.
clustersH <- clusterJH(df, distancia, 1:2, "average", 50, name)
```

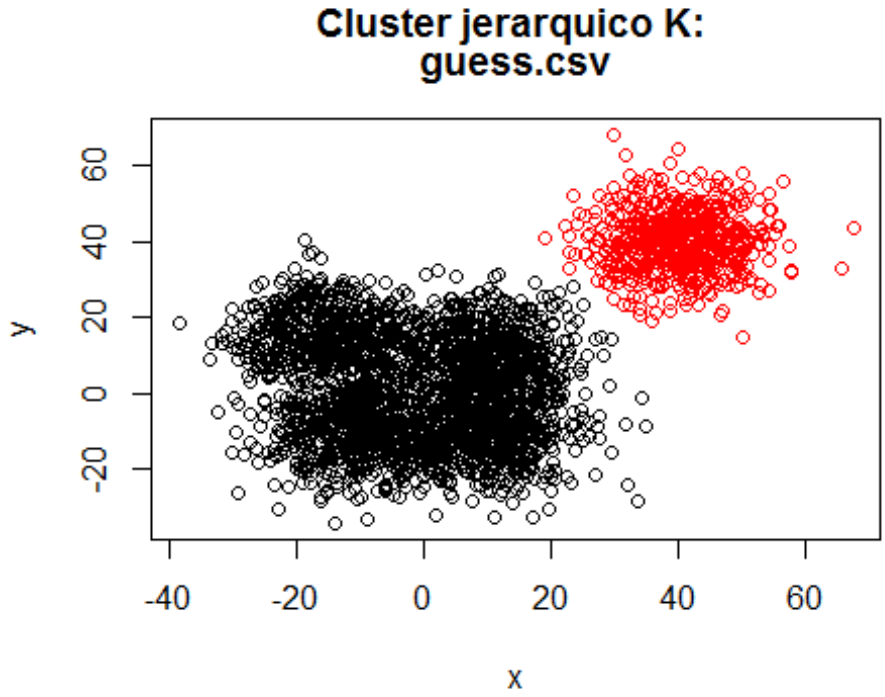



```
## [1] 1 2
```

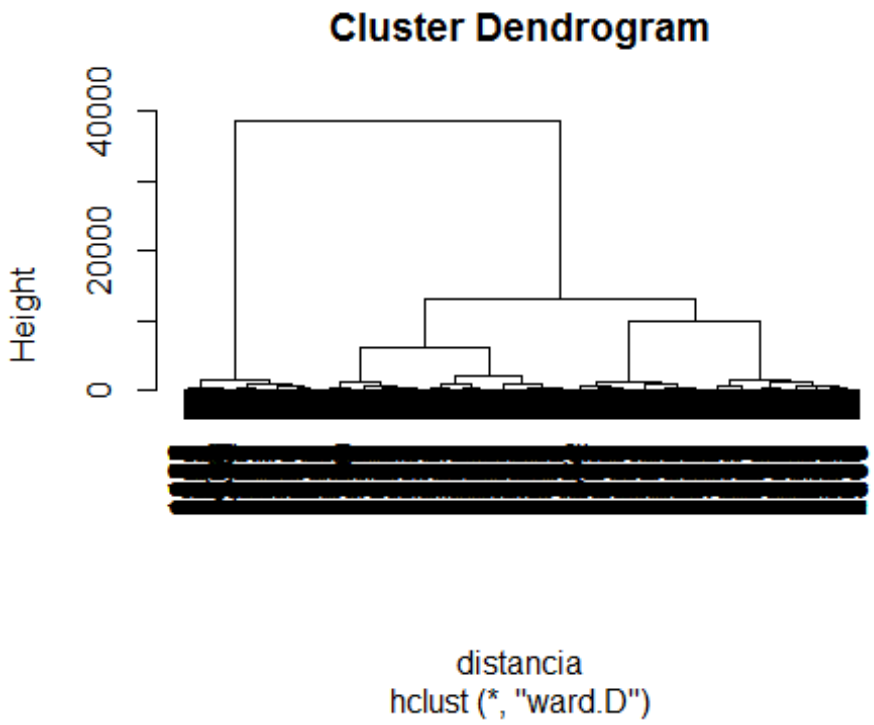


Método ward.D

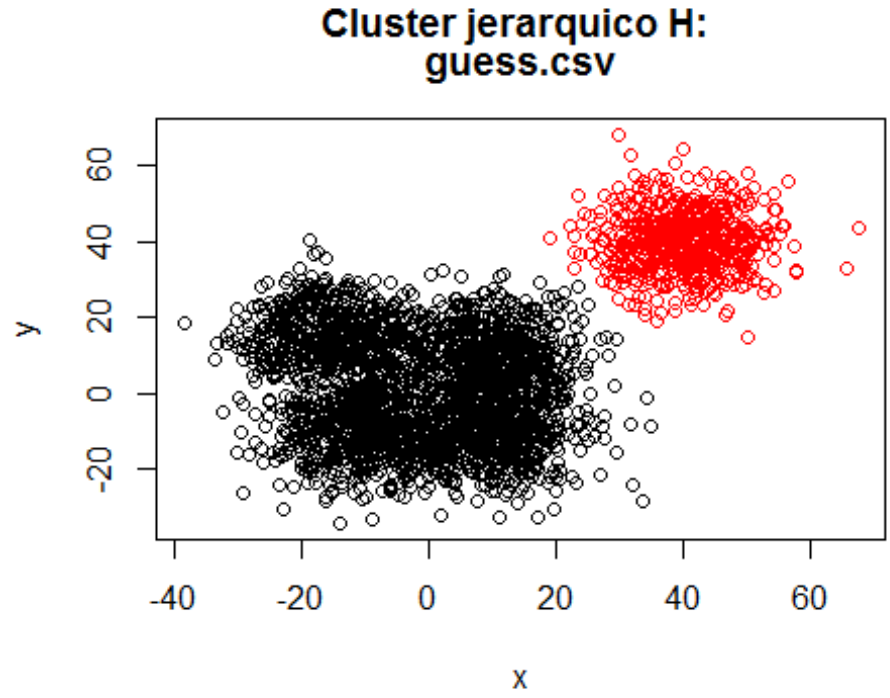
```r
#-------------------------------------------------------------------------
----------
#                                    METHOD ward.D
#-------------------------------------------------------------------------
----------
#Dado un número de clústers k determinar la altura requerida
#para que tengamos el número de clúster k.
clustersD <- clusterJD(df, distancia, 1:2, "ward.D", 2, name)
```




```r
#*********************************************************
#Dada una altura h (una medida de disimilaridad) determinar
#el número de clústers que se obtienen.
clustersH <- clusterJH(df, distancia, 1:2, "ward.D", 30000, name)
```


Cluster Dendrogram
Height
0
20000
40000
distancia
hclust (*, "ward.D")

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


Cluster jerarquico H:
guess.csv
y
-20
0
20
40
60
-40
-20
0
20
40
60
x

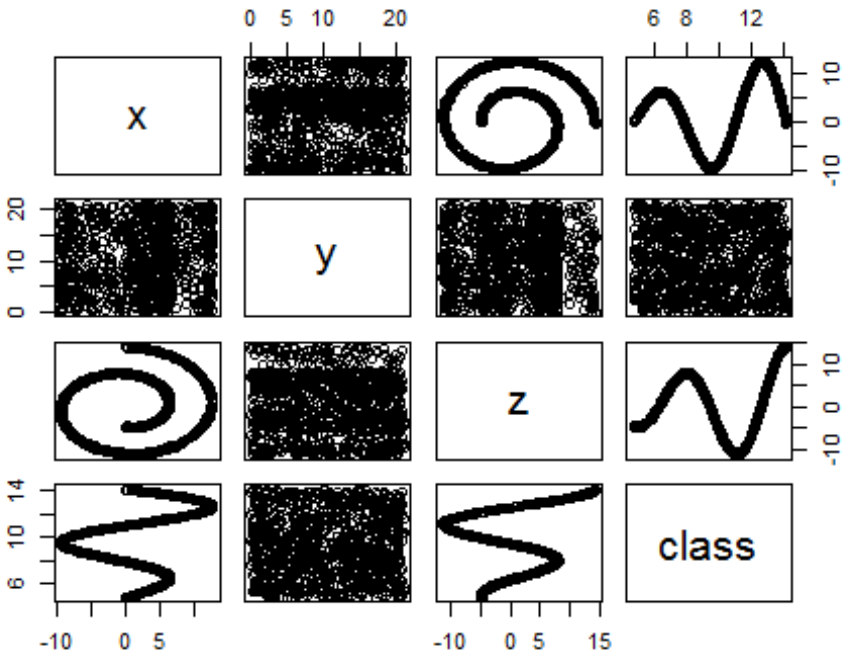
h.csv

- Preprocesamiento:

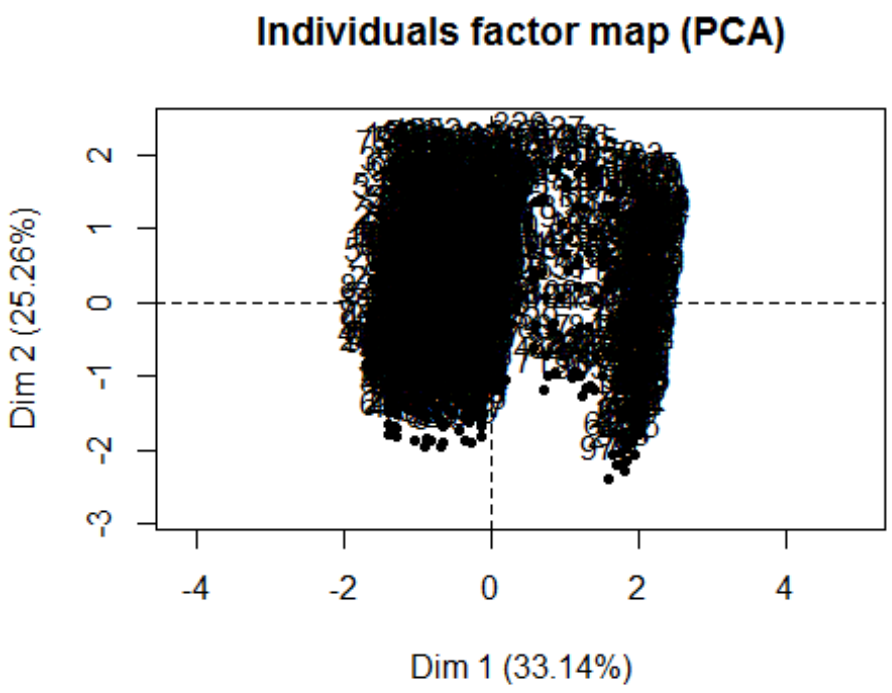
```
setwd("C:/Users/Eric/Desktop/AprendizajeNoSupervisado")
name = "h.csv"
library(rgl)
library('FactoMineR')
library(pROC)
#Lectura de datos.
df = read.csv(file =
"C:/Users/Eric/Desktop/AprendizajeNoSupervisado/data/h.csv", header = F)
#Modificamos el nombre de las columnas por comodidad.
colnames(df) <- c("x","y","z","class")

#Coloco las clases del 1:n
df$class = as.numeric(df$class)
if (min(df$class) == 0){
  df$class <- df$class + 1
}

#Ordenamos la columna clase
df <- df[ order(df$class), ]
```

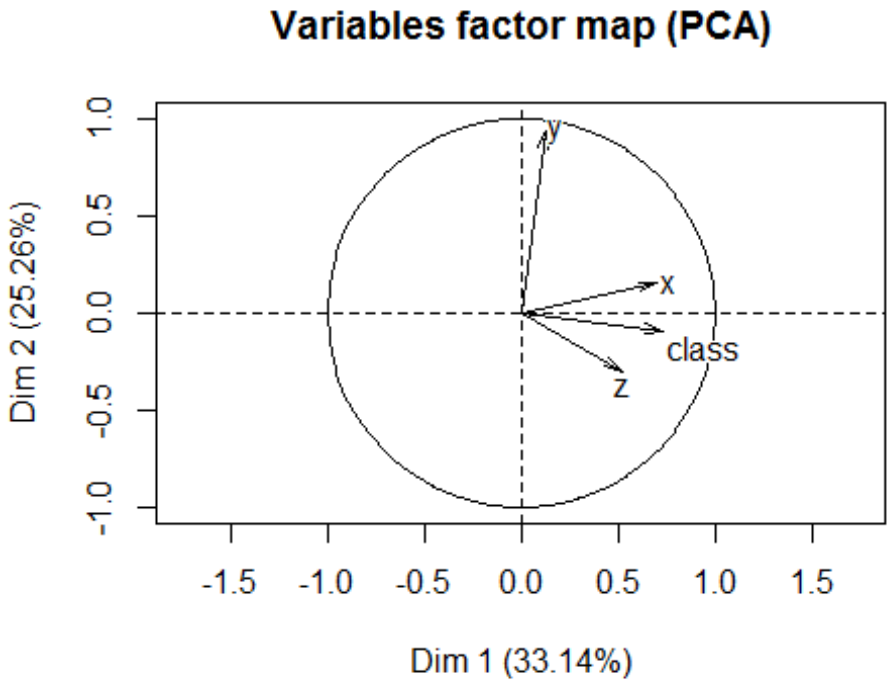
- Analisis exploratorio del dataset:

```
#*************************************************************************
************
#                    Analisis exploratorio del dataset
#*************************************************************************
************
#Podemos observar que hay 4 columnas.
head(df)

##             x         y         z    class
## 822  0.166535  7.195527 -4.798943 4.717535
## 758 -0.035730 20.303708 -4.651489 4.725427
## 541  0.000952 10.442965 -4.736662 4.737866
## 743  0.201462  4.929249 -4.533533 4.748772
## 100  0.165747 11.621006 -4.765143 4.756643
## 853  0.282215  5.946968 -4.679441 4.760955

#Matriz de dispersion
plot(df)
```

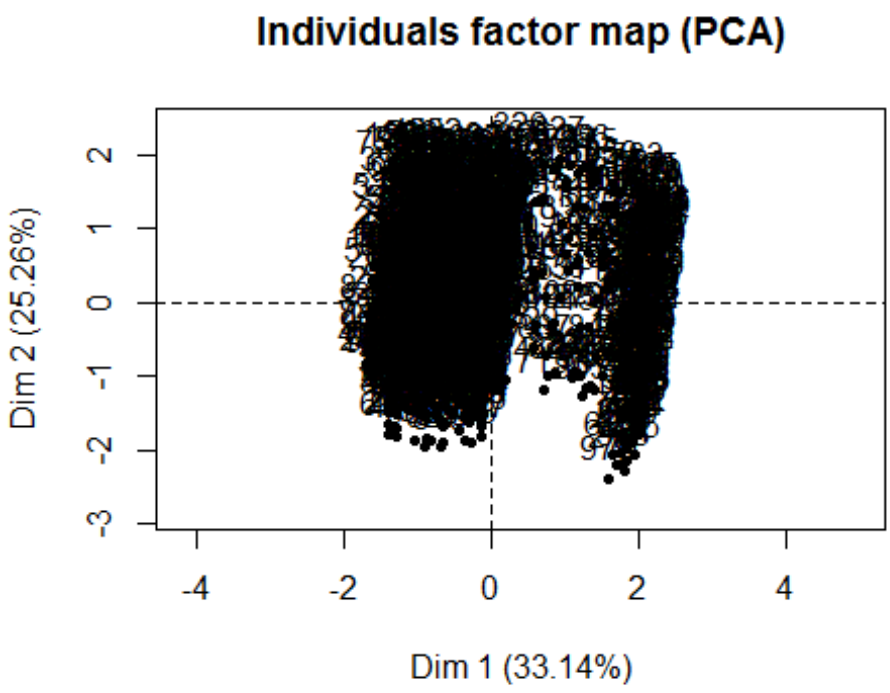



```
PCA <- PCA(df)
```

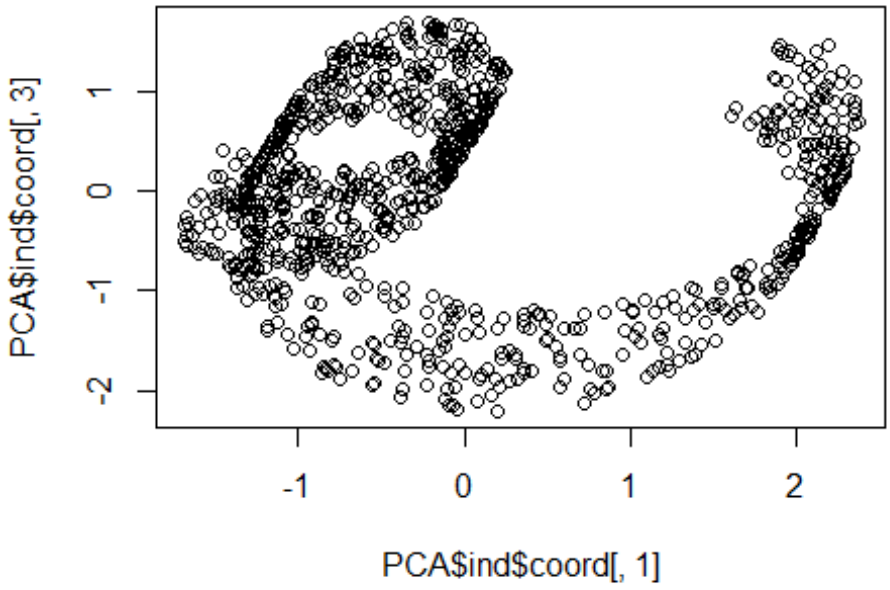

Individuals factor map (PCA)
Dim 2 (25.26%)
Dim 1 (33.14%)
-4
-2
0
2
4
-3
-2
-1
0
1
2


Variables factor map (PCA)
Dim 2 (25.26%)
Dim 1 (33.14%)
y
x
class
z
-1.5
-1.0
-0.5
0.0
0.5
1.0
1.5
-1.0
-0.5
0.0
0.5
1.0

```
plot(PCA)
```




```r
plot(PCA$ind$coord[,1],PCA$ind$coord[,3])
```




```r
#Grafico
palette(rainbow(14))
```

```r
plot3d(df$x, df$y, df$z, type = "s", size = 2, col = df$class)
#Podemos observar 11 conglomerados
```

Reglas para asignar clases:

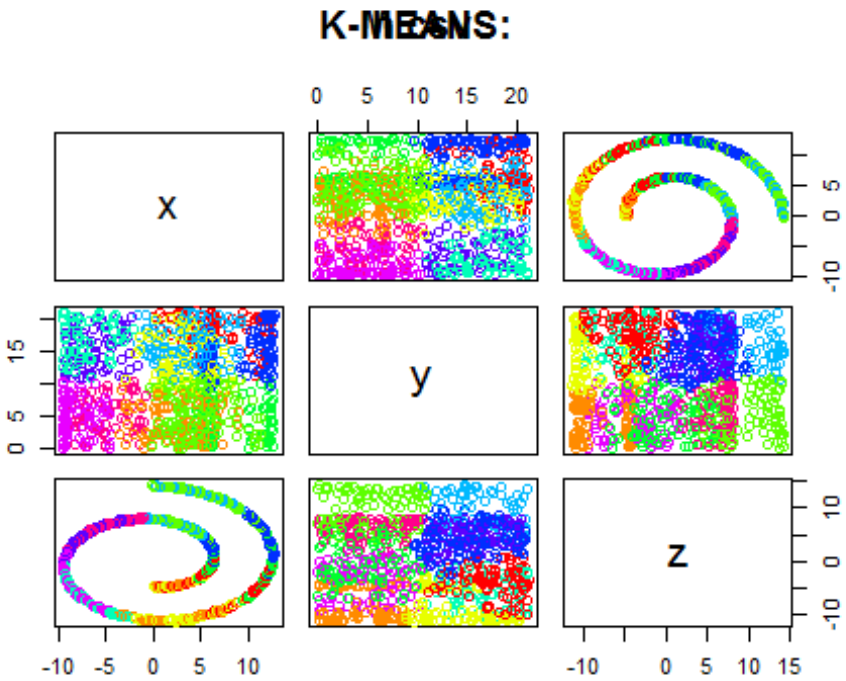
El criterio para asignar estas clases fue utilizando el graficador 3D donde aparentemente existen 11 clases distintas, por lo tanto decidí 11 clases en total.

```r
#**********************REGLA PARA ASIGNAR CLASES**************************
reglas <- function(x){
  if (x < 4.99735){
    return(1)
  }else if (x < 5.989202){
    return(2)
  }else if (x < 6.994868){
    return(3)
  }else if (x < 7.997107){
    return(4)
  }else if (x < 8.992769){
    return(5)
  }else if (x < 9.987271){
    return(6)
  }else if (x < 10.99652){
    return(7)
  }else if (x < 11.99878){
    return(8)
  }else if (x < 12.99898){
    return(9)
  }else if (x < 13.98096){
    return(10)
  }else{
    return(11)
  }
}

for (x in 1:nrow(df)) {

  df$class[x] <- reglas(df$class[x])
}
#Observamos cuantos elementos hay de cada clase.
table(df$class)

##
##   1   2   3   4   5   6   7   8   9  10  11
##  29 110 104 115 116  98  98 103  95 118  14

#1   2   3   4   5   6   7   8   9  10  11
#29 110 104 115 116  97  98 103  95 118  14
```

```
palette(rainbow(length(unique(df$class))))
plot3d(df$x, df$y, df$z, type = "s",size = 2, col = df$class)
```

K-medias:

```
#*********************************************************************************
************
                                        #K-MEDIAS
#*********************************************************************************
************
#Aplicamos k=11 ya que identificamos 11 conglomerados.
#kmedias(Dataframe, Columnas,K)
modeloK <- kmedias(df, 1:3, 11, name)
```



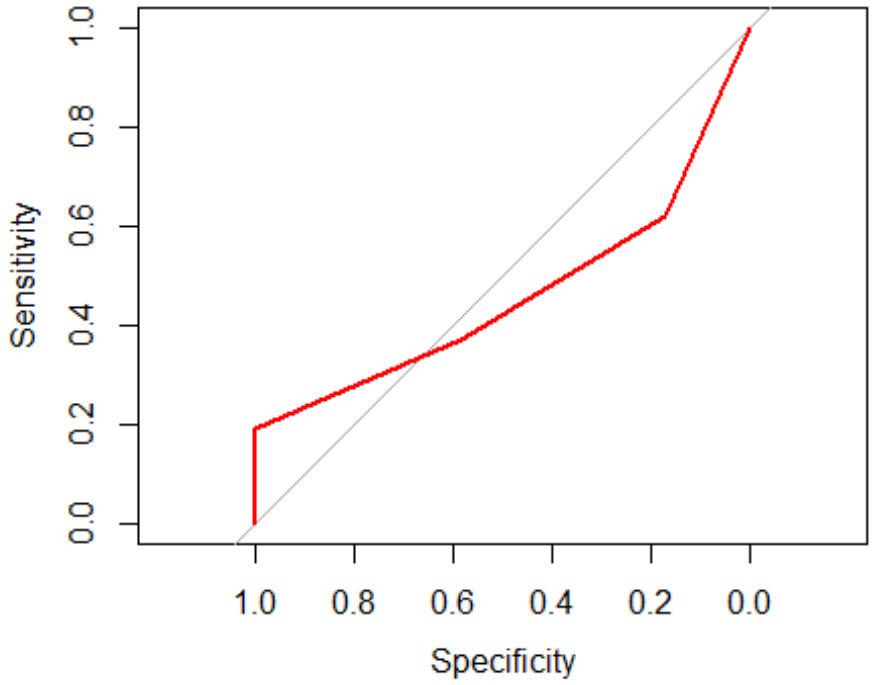

```
plot3d(df$x, df$y, df$z, type = "s",size = 2, col = modeloK$cluster)
#Generamos la matriz de confusion
MatrixConfusionK <- matrizconfusion(df$class,modeloK$cluster)
MatrixConfusionK

##      Cluster
## Clase  1  2  3  4  5  6  7  8  9 10 11
##    1  12  5  0  0  0  0  0 12  0  0  0
##    2  27 42  0  0  0  0  0 20 21  0  0
##    3   0 20 41  0  0  0  0  0 36  7  0
##    4   0  0  6 49  0  0  0  0  0 55  5
##    5   0  0  0  2 59  0  0  0  0  0 55
##    6   0  0  0  0 27 42 20  0  0  0  9
```

```
##    7  19  0  0  0  0 34 39  6  0  0  0
##    8  36 16  0  0  0  0  0 48  3  0  0
##    9   0 22 36  0  0  0  0  0 37  0  0
##    10  0  0 26 31  0  0  0  0  3 58  0
##    11  0  0  0  7  0  0  0  0  0  7  0
```

```
#Calculamos la precision del modelo
PrecisionK <- precision(MatrixConfusionK)
PrecisionK
```

```
## [1] 0.427
```

```
#Generamos la curva de ROC
modeloKROC <- roc(df$class, modeloK$cluster)
plot(modeloKROC,type="l",col="red")
```



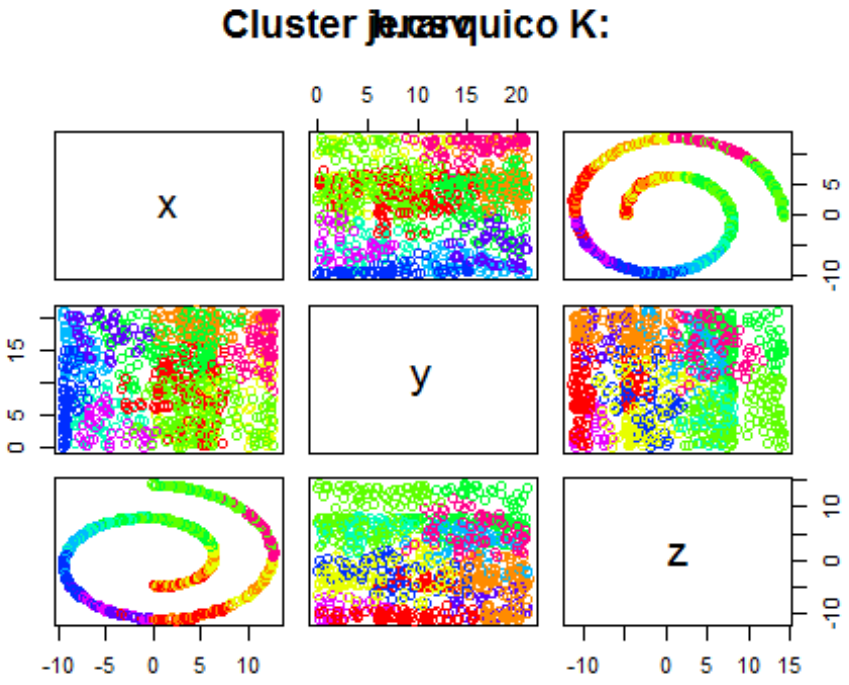

```
##
## Call:
## roc.default(response = df$class, predictor = modeloK$cluster)
##
## Data: modeloK$cluster in 29 controls (df$class 1) < 110 cases (df$class
2).
## Area under the curve: 0.4611
```

Clusters jerárquicos:

```
#*************************************************************************************
************
#                                Cluster Jerarquicos
#*************************************************************************************
************
#Copy del dataset
datos = df
#Elimino la columna clase para realizar aprendizaje no supervisado.
datos$class <- NULL
#Convierto el dataframe en una matriz
datos= as.matrix(datos)
#Calculamos la matriz de distancia
distancia = dist(datos)
```

Método complete:

```
#-------------------------------------------------------------------------------
----------
#                                METHOD COMPLETE
#-------------------------------------------------------------------------------
----------
#Dado un número de clústers k determinar la altura requerida
#para que tengamos el número de clúster k.
clustersD <- clusterJD(df, distancia, 1:3, "complete", 11, name)
```



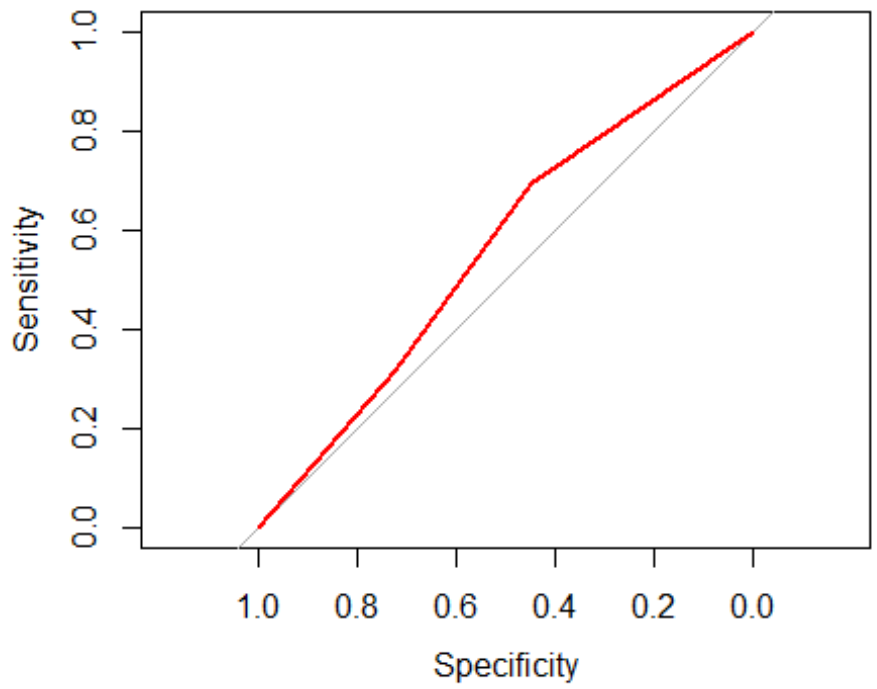
```r
#Generamos la matriz de confusion
plot3d(df$x, df$y, df$z, type = "s",size = 2, col = clustersD)
MatrixConfusionCJDC <- matrizconfusion(df$class, clustersD)
MatrixConfusionCJDC
```

```
##       Cluster
## Clase  1  2  3  4  5  6  7  8  9 10 11
##     1  0  8  8  0  0  0  0 13  0  0  0
##     2  0 41 35  0  0  0  0 34  0  0  0
##     3  0 27 43 21  0  0  0  0  0 13  0
##     4  0  0  1 55  2  0  0  0  0 57  0
##     5 30  0  0 38 42  0  0  0  0  6  0
##     6 35  0  0  0  8 52  3  0  0  0  0
##     7  0  0  0  0  0 13 39  9  0  0 37
##     8  0 33  8  0  0  0  0 61  0  0  1
##     9  0 17 35  0  0  0  0  0 31 12  0
##     10 0  0  0 25  0  0  0  0 31 62  0
##     11 0  0  0  6  0  0  0  0  0  8  0
```

```r
#Calculamos la precision del modelo
PrecisionDC <- precision(MatrixConfusionCJDC)
PrecisionDC
```

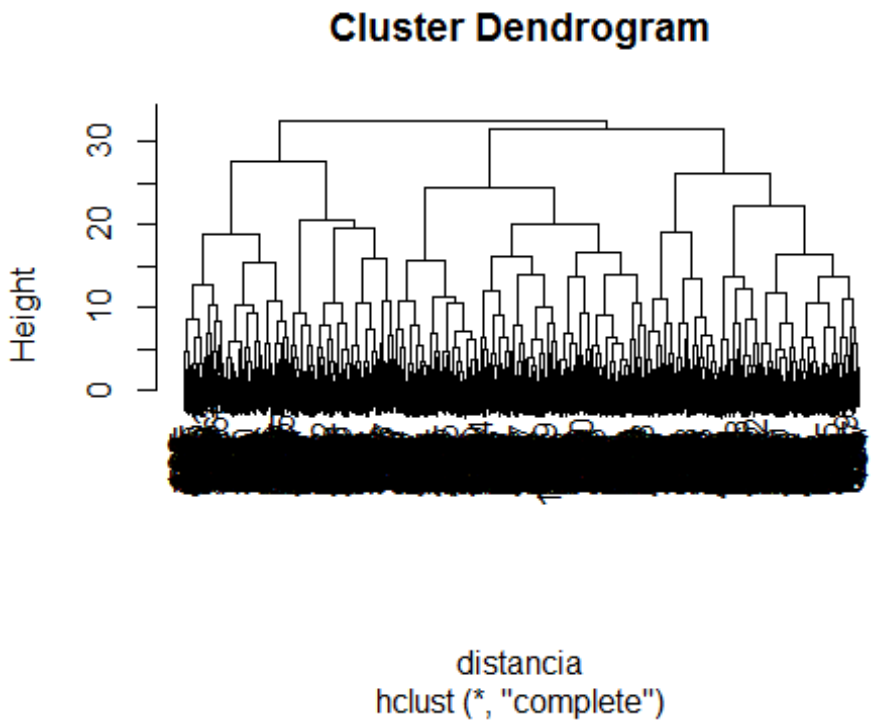
```
## [1] 0.426
```

```r
#Generamos la curva de ROC
modeloDC <- roc(df$class, clustersD)
plot(modeloDC,type="l",col="red")
```

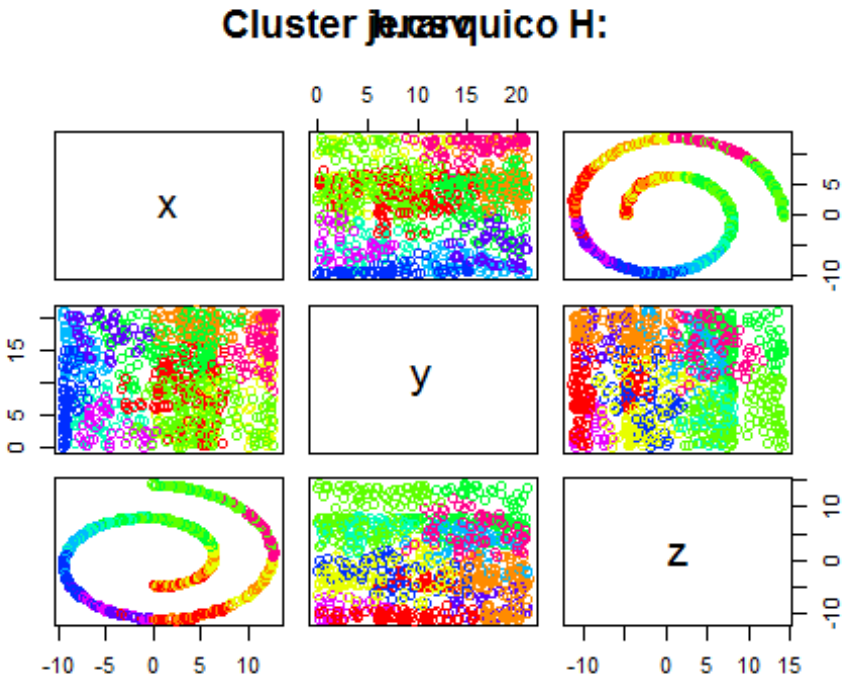


```
##
## Call:
## roc.default(response = df$class, predictor = clustersD)
##
## Data: clustersD in 29 controls (df$class 1) < 110 cases (df$class 2).
## Area under the curve: 0.5621
```

```
#*********************************************************
#Dada una altura h (una medida de disimilaridad) determinar
#el número de clústers que se obtienen.
clustersH <- clusterJH(df, distancia, 1:3, "complete", 19, name)
```




```
##  [1]  1  2  3  4  5  6  7  8  9 10 11
```



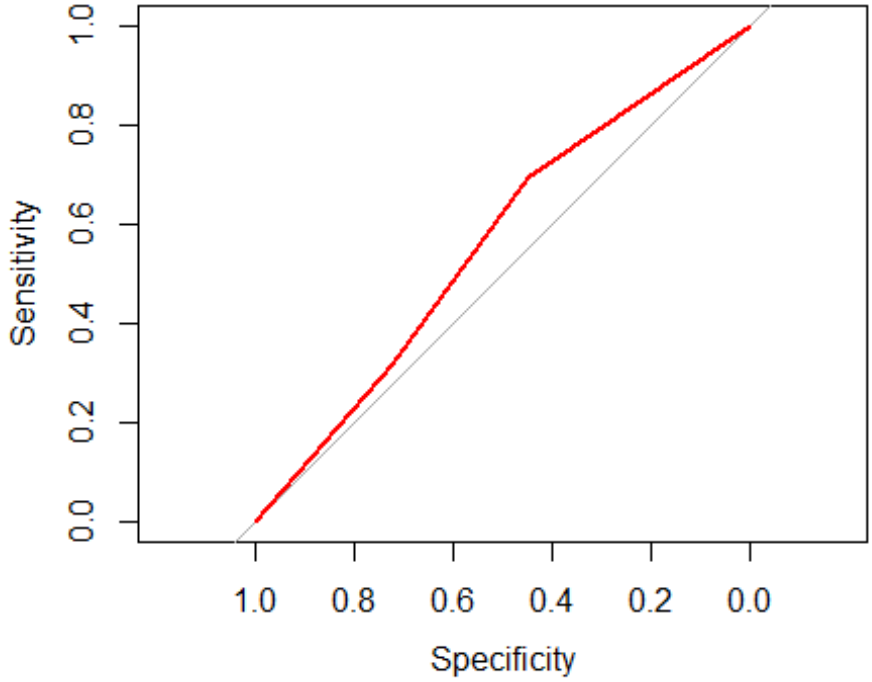

```
plot3d(df$x, df$y, df$z, type = "s",size = 2, col = clustersH)
#Generamos la matriz de confusion
MatrixConfusionCJHC <- matrizconfusion(df$class, clustersH)
MatrixConfusionCJHC
```

```
##       Cluster
## Clase  1  2  3  4  5  6  7  8  9 10 11
##     1  0  8  8  0  0  0  0 13  0  0  0
##     2  0 41 35  0  0  0  0 34  0  0  0
##     3  0 27 43 21  0  0  0  0  0 13  0
##     4  0  0  1 55  2  0  0  0  0 57  0
##     5 30  0  0 38 42  0  0  0  0  6  0
##     6 35  0  0  0  8 52  3  0  0  0  0
##     7  0  0  0  0  0 13 39  9  0  0 37
##     8  0 33  8  0  0  0  0 61  0  0  1
##     9  0 17 35  0  0  0  0  0 31 12  0
##     10 0  0  0 25  0  0  0  0 31 62  0
##     11 0  0  0  6  0  0  0  0  0  8  0
```

```
#Calculamos la precision del modelo
PrecisionHC <- precision(MatrixConfusionCJHC)
PrecisionHC
```

```
## [1] 0.426
```

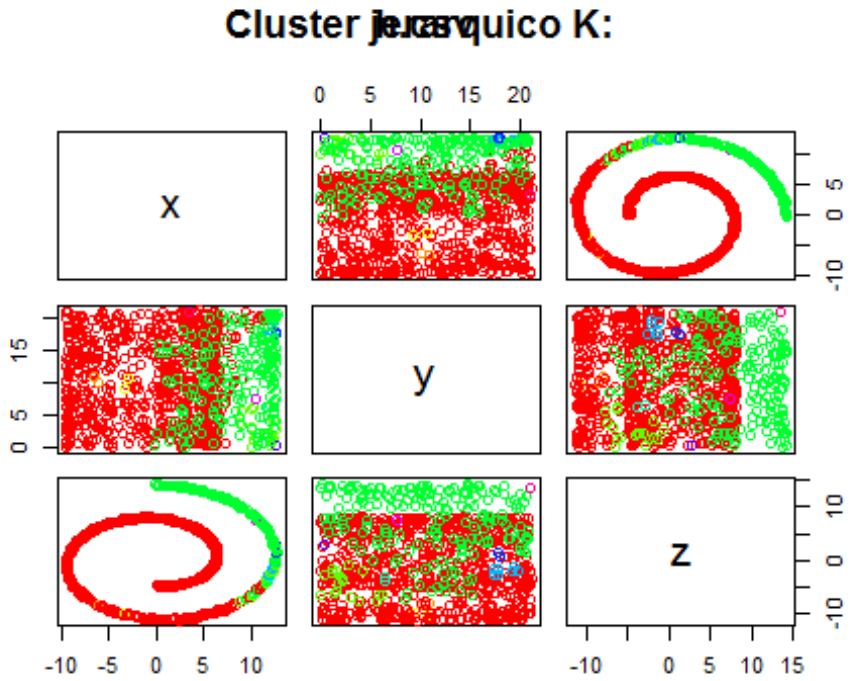
```
#Generamos la curva de ROC
modeloHC <- roc(df$class, clustersH)
plot(modeloHC,type="l",col="red")
```




```
##
## Call:
## roc.default(response = df$class, predictor = clustersH)
##
## Data: clustersH in 29 controls (df$class 1) < 110 cases (df$class 2).
## Area under the curve: 0.5621
```

Método single:

```
#-------------------------------------------------------------------------
----------
#                                METHOD SINGLE
#-------------------------------------------------------------------------
----------
#Dado un número de clústers k determinar la altura requerida
#para que tengamos el número de clúster k.
clustersD <- clusterJD(df, distancia, 1:3, "single", 11, name)
```



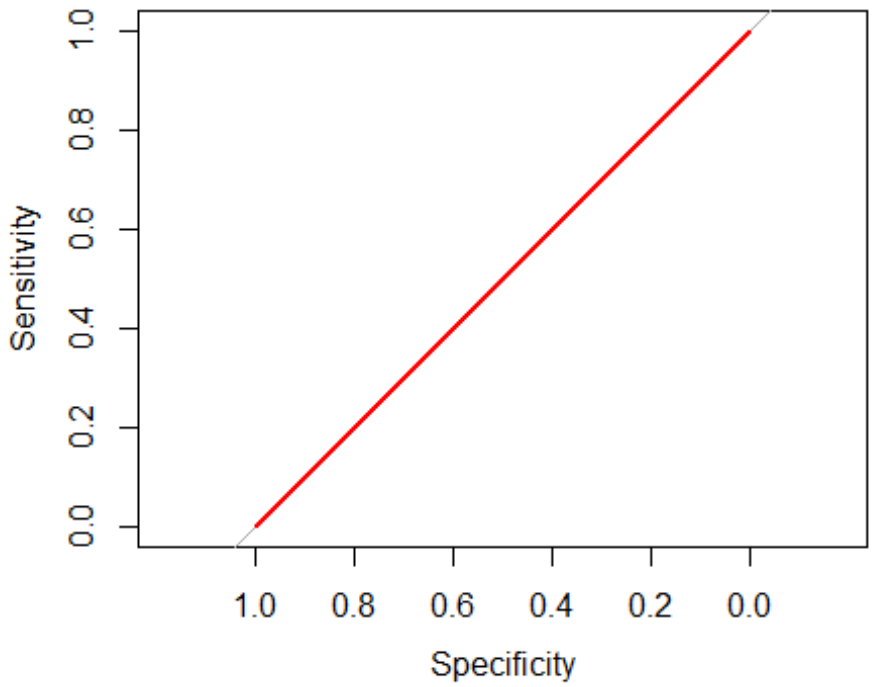

```
plot3d(df$x, df$y, df$z, type = "s",size = 2, col = clustersD)
#Generamos la matriz de confusion
MatrixConfusionCJDS <- matrizconfusion(df$class, clustersD)
MatrixConfusionCJDS
```

```
##       Cluster
## Clase   1   2   3   4   5   6   7   8   9  10  11
##     1   0   0   0   0  29   0   0   0   0   0   0
##     2   0   0   0   0 110   0   0   0   0   0   0
##     3   0   0   0   0 104   0   0   0   0   0   0
##     4   0   0   0   0 115   0   0   0   0   0   0
##     5   0   0   0   0 116   0   0   0   0   0   0
##     6   0   0   0   0  98   0   0   0   0   0   0
##     7   2   0   0   0  94   0   2   0   0   0   0
##     8   0   0   0   0  97   0   0   0   4   2   0
##     9   0   2   7   2   7   1   0   0  15  61   0
##     10  0   0   0   0   0   0   0   1   0 116   1
##     11  0   0   0   0   0   0   0   0   0  14   0
```

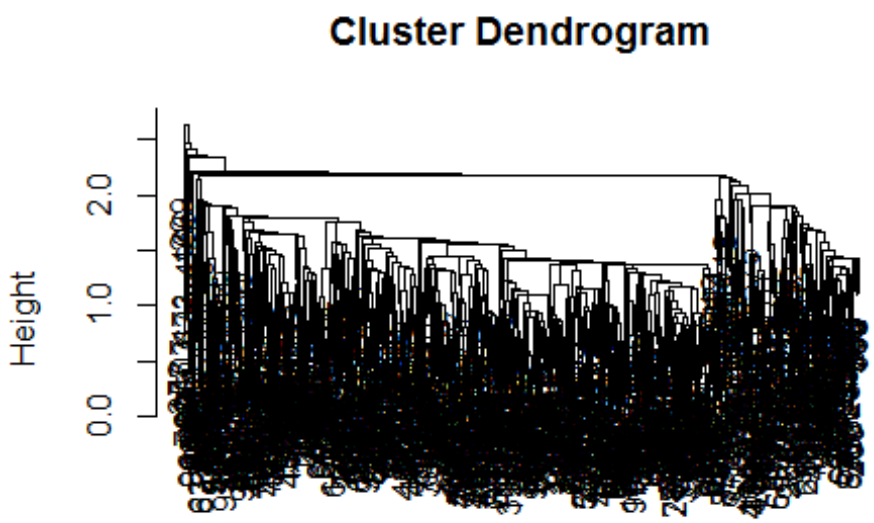
```
#Calculamos la precision del modelo
PrecisionDS <- precision(MatrixConfusionCJDS)
PrecisionDS
```

```
## [1] 0.249
```

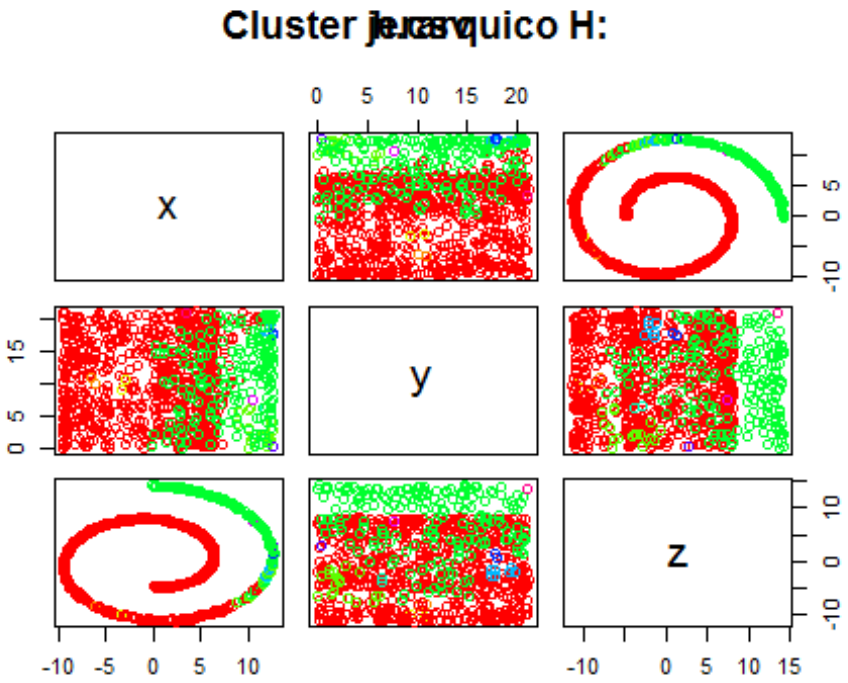
```
#Generamos la curva de ROC
modeloDS <- roc(df$class, clustersD)
plot(modeloDS,type="l",col="red")
```




```
##
## Call:
## roc.default(response = df$class, predictor = clustersD)
##
## Data: clustersD in 29 controls (df$class 1) < 110 cases (df$class 2).
## Area under the curve: 0.5

#*********************************************************
#Dada una altura h (una medida de disimilaridad) determinar
#el número de clústers que se obtienen.
clustersH <- clusterJH(df, distancia, 1:3, "single", 2.145, name)
```




```
##  [1]  1  2  3  4  5  6  7  8  9 10 11
```




```
plot3d(df$x, df$y, df$z, type = "s",size = 2, col = clustersH)
#Generamos la matriz de confusion
```

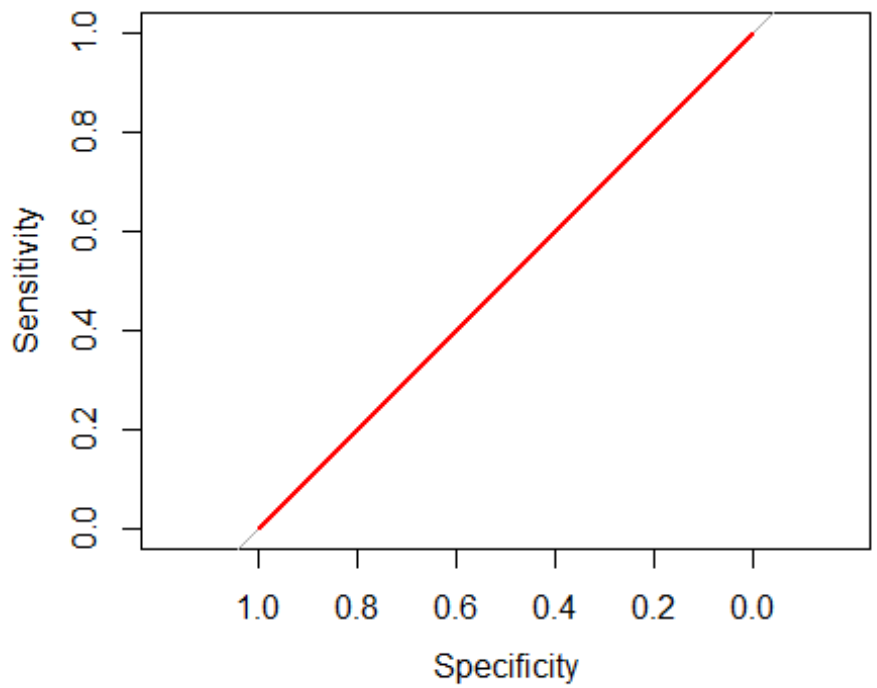
```r
MatrixConfusionCJHS <- matrizconfusion(df$class, clustersH)
MatrixConfusionCJHS
```

```
##       Cluster
## Clase   1   2   3   4   5   6   7   8   9  10  11
##     1   0   0   0   0  29   0   0   0   0   0   0
##     2   0   0   0   0 110   0   0   0   0   0   0
##     3   0   0   0   0 104   0   0   0   0   0   0
##     4   0   0   0   0 115   0   0   0   0   0   0
##     5   0   0   0   0 116   0   0   0   0   0   0
##     6   0   0   0   0  98   0   0   0   0   0   0
##     7   2   0   0   0  94   0   2   0   0   0   0
##     8   0   0   0   0  97   0   0   0   4   2   0
##     9   0   2   7   2   7   1   0   0  15  61   0
##     10  0   0   0   0   0   0   0   1   0 116   1
##     11  0   0   0   0   0   0   0   0   0  14   0
```

```r
#Calculamos la precision del modelo
PrecisionHS <- precision(MatrixConfusionCJHS)
PrecisionHS
```

```
## [1] 0.249
```

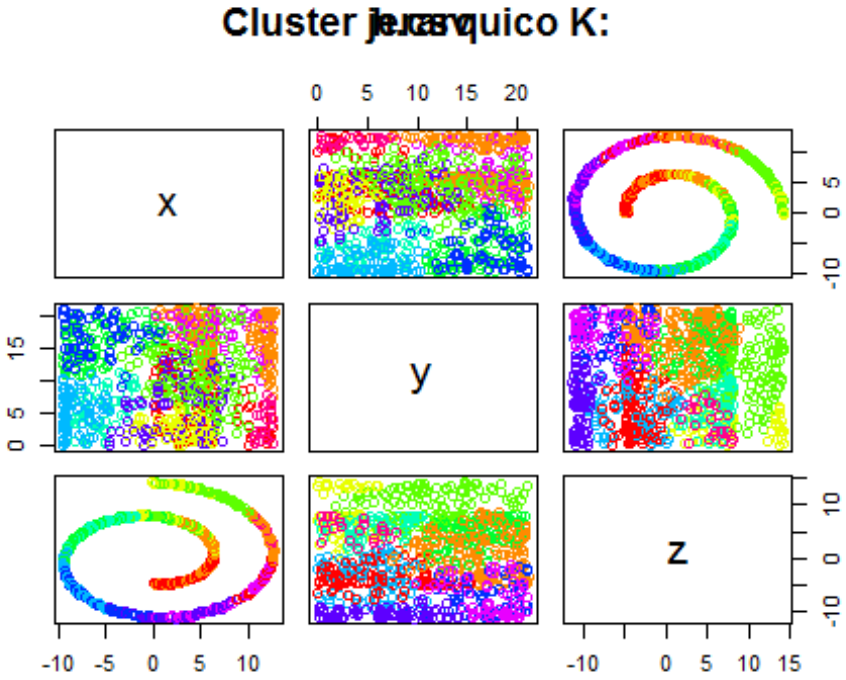
```r
#Generamos la curva de ROC
modeloHS <- roc(df$class, clustersH)
plot(modeloHS,type="l",col="red")
```



```
##
## Call:
## roc.default(response = df$class, predictor = clustersH)
##
## Data: clustersH in 29 controls (df$class 1) < 110 cases (df$class 2).
## Area under the curve: 0.5
```

Método average:

```
#-------------------------------------------------------------------------------
----------
#                                   METHOD AVERAGE
#-------------------------------------------------------------------------------
----------
#Dado un número de clústers k determinar la altura requerida
#para que tengamos el número de clúster k.
clustersD <- clusterJD(df, distancia, 1:3, "average", 11, name)
```




```
plot3d(df$x, df$y, df$z, type = "s",size = 2, col = clustersD)
#Generamos la matriz de confusion
MatrixConfusionCJDA <- matrizconfusion(df$class, clustersD)
MatrixConfusionCJDA
```

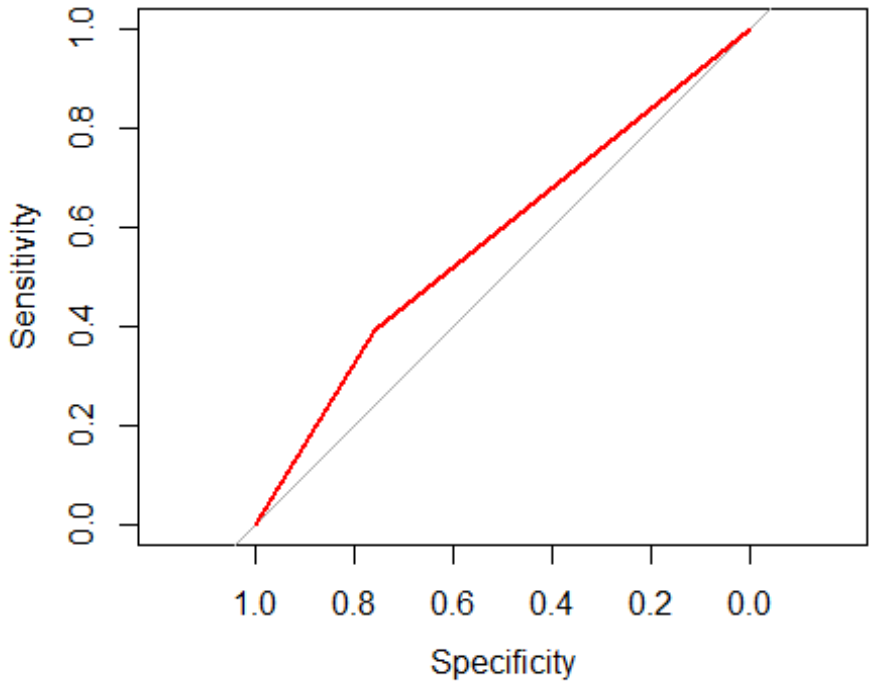
```
##       Cluster
## Clase  1  2  3  4  5  6  7  8  9 10 11
##    1   0 22  7  0  0  0  0  0  0  0  0
##    2   0 67 43  0  0  0  0  0  0  0  0
```

```
##    3   0 25 60 17  0  0  0  0  0  2  0
##    4   1  0 25 28 14  0  0  0  0 47  0
##    5  49  0  0  8 54  0  0  0  0  5  0
##    6   9  0  0  0 31 42 16  0  0  0  0
##    7   0  0  0  0  0 32 45 21  0  0  0
##    8   0  5  0  0  0  0  2 60 36  0  0
##    9   0 19 38  0  0  0  0  0 26  0 12
##    10  0  0 18 14  0  0  0  0  0 75 11
##    11  0  0  0  5  0  0  0  0  0  9  0
```

```r
#Calculamos la precision del modelo
PrecisionDA <- precision(MatrixConfusionCJDA)
PrecisionDA
```

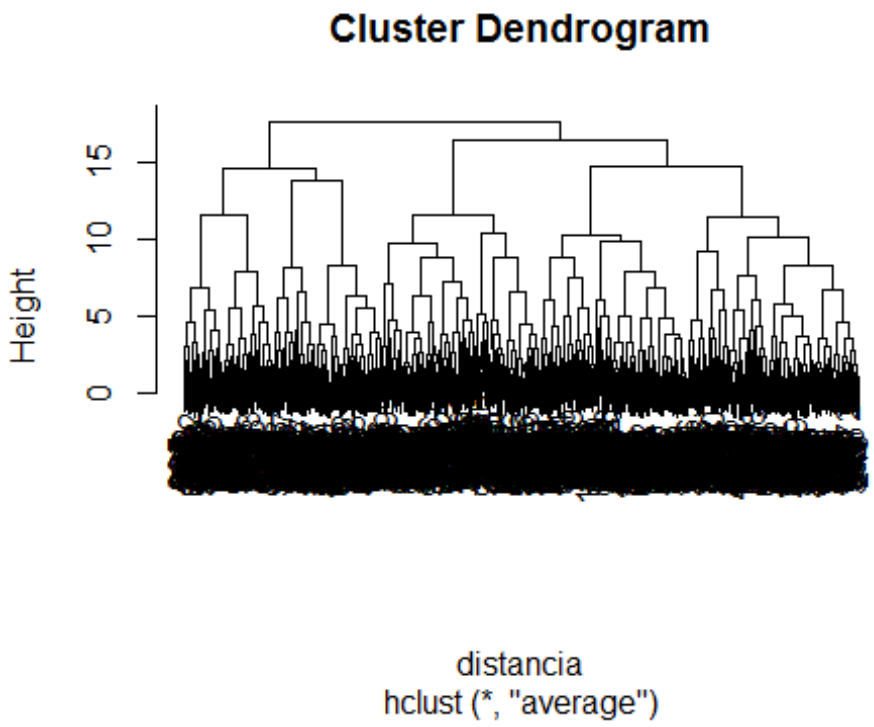
```
## [1] 0.457
```

```r
#Generamos la curva de ROC
modeloDA <- roc(df$class, clustersD)
plot(modeloDA,type="l",col="red")
```

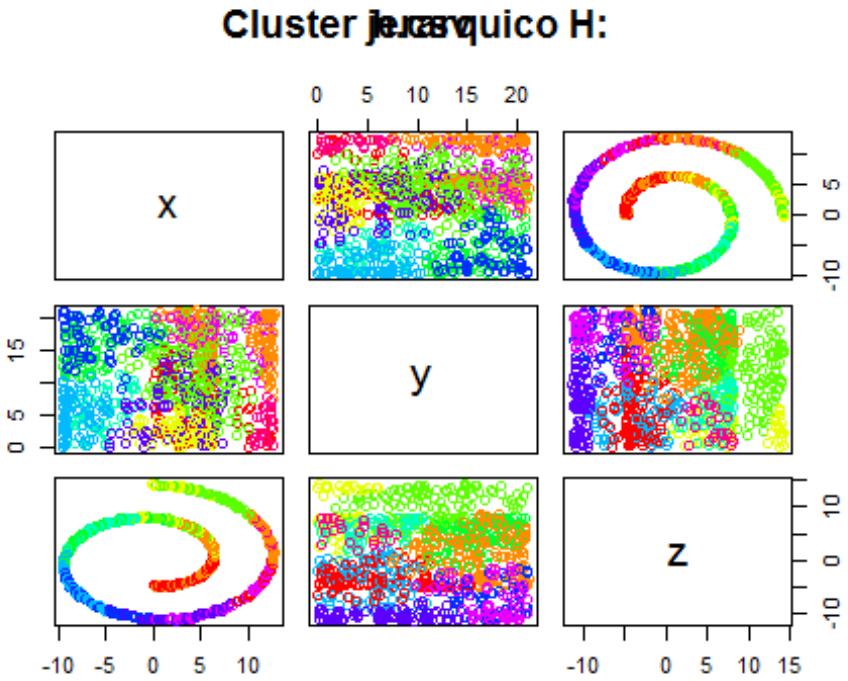



```
##
## Call:
## roc.default(response = df$class, predictor = clustersD)
##
## Data: clustersD in 29 controls (df$class 1) < 110 cases (df$class 2).
## Area under the curve: 0.5748
```

```r
#*********************************************************
#Dada una altura h (una medida de disimilaridad) determinar
#el número de clústers que se obtienen.
clustersH <- clusterJH(df, distancia, 1:3, "average", 10.2, name)
```




```
##  [1]  1  2  3  4  5  6  7  8  9 10 11
```



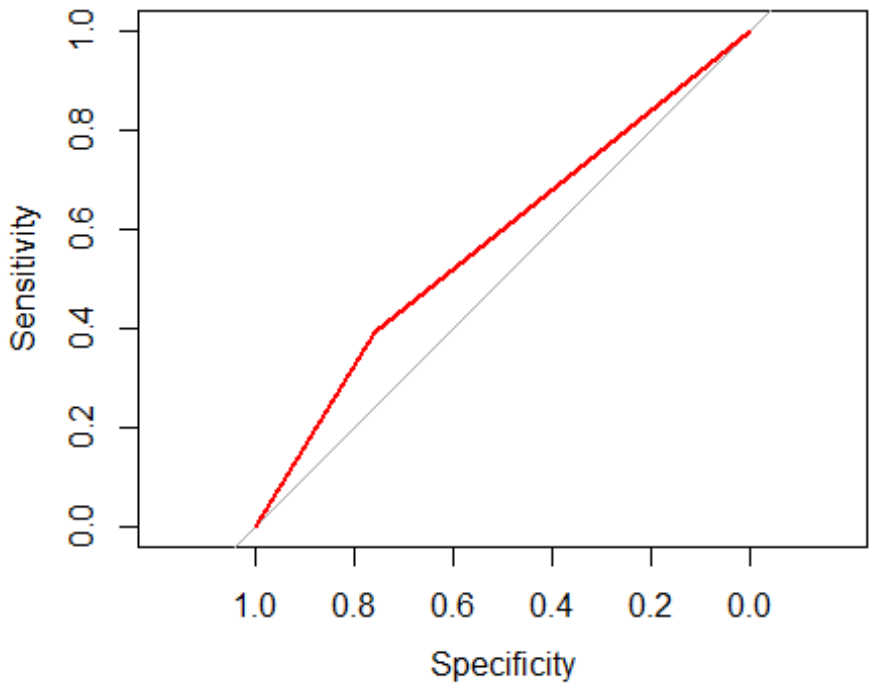

```
plot3d(df$x, df$y, df$z, type = "s",size = 2, col = clustersH)
#Generamos la matriz de confusion
MatrixConfusionCJHA <- matrizconfusion(df$class, clustersH)
MatrixConfusionCJHA
```

```
##       Cluster
## Clase  1  2  3  4  5  6  7  8  9 10 11
##     1  0 22  7  0  0  0  0  0  0  0  0
##     2  0 67 43  0  0  0  0  0  0  0  0
##     3  0 25 60 17  0  0  0  0  0  2  0
##     4  1  0 25 28 14  0  0  0  0 47  0
##     5 49  0  0  8 54  0  0  0  0  5  0
##     6  9  0  0  0 31 42 16  0  0  0  0
##     7  0  0  0  0  0 32 45 21  0  0  0
##     8  0  5  0  0  0  0  2 60 36  0  0
##     9  0 19 38  0  0  0  0  0 26  0 12
##     10 0  0 18 14  0  0  0  0  0 75 11
##     11 0  0  0  5  0  0  0  0  0  9  0
```

```
#Calculamos la precision del modelo
PrecisionHA <- precision(MatrixConfusionCJHA)
PrecisionHA
```

```
## [1] 0.457
```

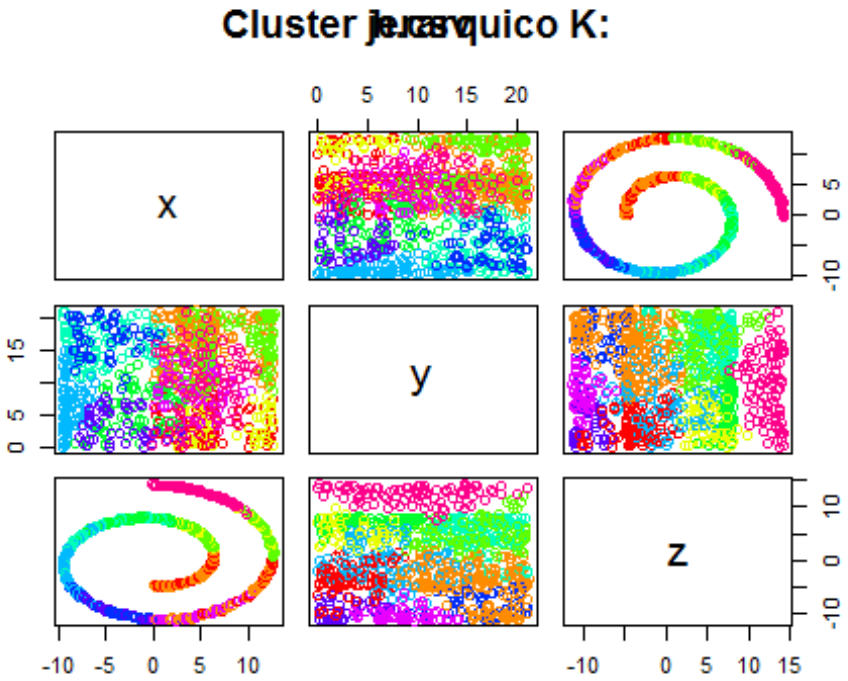
```
#Generamos la curva de ROC
modeloHA <- roc(df$class, clustersH)
plot(modeloHA,type="l",col="red")
```




```
##
## Call:
## roc.default(response = df$class, predictor = clustersH)
##
## Data: clustersH in 29 controls (df$class 1) < 110 cases (df$class 2).
## Area under the curve: 0.5748
```

Método ward.D

```
#-------------------------------------------------------------------------
----------
#                              METHOD ward.D
#-------------------------------------------------------------------------
----------
#Dado un número de clústers k determinar la altura requerida
#para que tengamos el número de clúster k.
clustersD <- clusterJD(df, distancia, 1:3, "ward.D", 11, name)
```



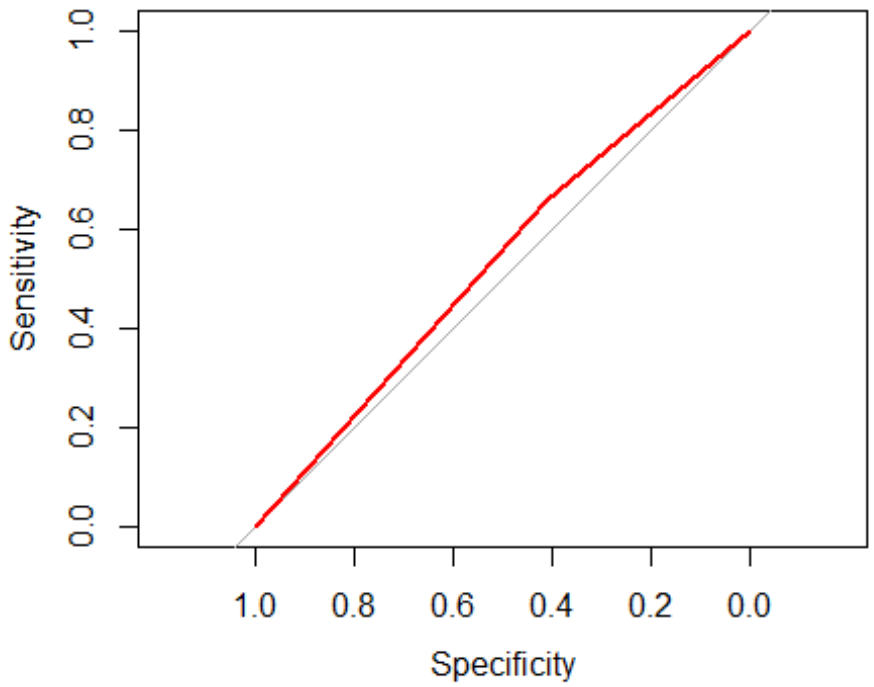

```
plot3d(df$x, df$y, df$z, type = "s",size = 2, col = clustersD)
#Generamos la matriz de confusion
MatrixConfusionCJDW <- matrizconfusion(df$class, clustersD)
MatrixConfusionCJDW
```

```
##       Cluster
## Clase  1  2  3  4  5  6  7  8  9 10 11
##     1  0 17 12  0  0  0  0  0  0  0  0
##     2  0 72 38  0  0  0  0  0  0  0  0
##     3  0 38 22 30  0  0  0  0 14  0  0
##     4 31  0  0 44 19  0  0  0 21  0  0
##     5 45  0  0  0 57 14  0  0  0  0  0
##     6  1  0  0  0 27 66  4  0  0  0  0
##     7  0  0  0  0  0  9 45  1  0  0 43
##     8  0 37 10  0  0  0  2 49  0  0  5
##     9  0 31 20 31  0  0  0  1 12  0  0
##     10 0  0  0 25  0  0  0  0 14 79  0
##     11 0  0  0  0  0  0  0  0  0 14  0
```

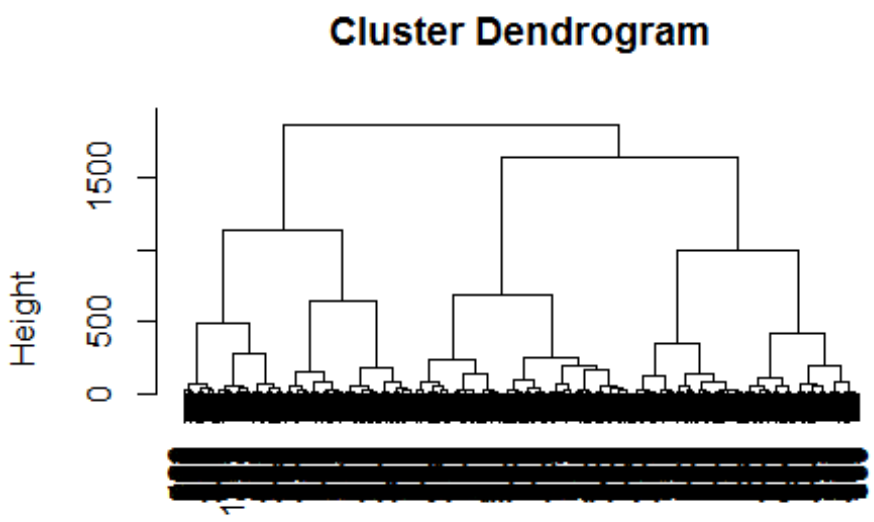
```
#Calculamos la precision del modelo
PrecisionDW <- precision(MatrixConfusionCJDW)
PrecisionDW
```

```
## [1] 0.446
```

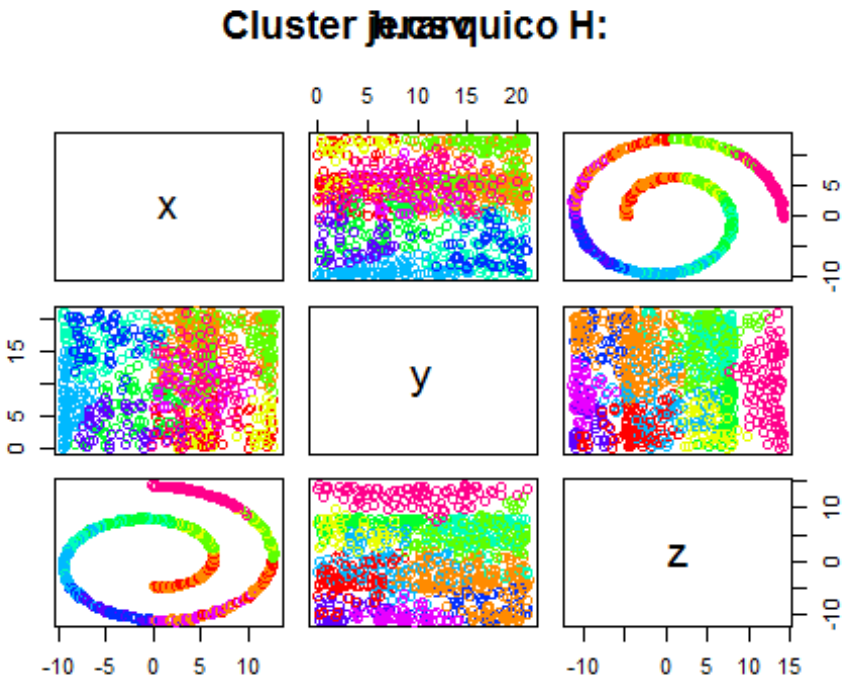
```
#Generamos la curva de ROC
modeloDW <- roc(df$class, clustersD)
plot(modeloDW,type="l",col="red")
```




```
##
## Call:
## roc.default(response = df$class, predictor = clustersD)
##
## Data: clustersD in 29 controls (df$class 1) < 110 cases (df$class 2).
## Area under the curve: 0.5342

#*********************************************************
#Dada una altura h (una medida de disimilaridad) determinar
#el número de clústers que se obtienen.
clustersH <- clusterJH(df, distancia, 1:3, "ward.D", 280, name)
```




```
##  [1]  1  2  3  4  5  6  7  8  9 10 11
```




```
plot3d(df$x, df$y, df$z, type = "s",size = 2, col = clustersH)
#Generamos la matriz de confusion
```

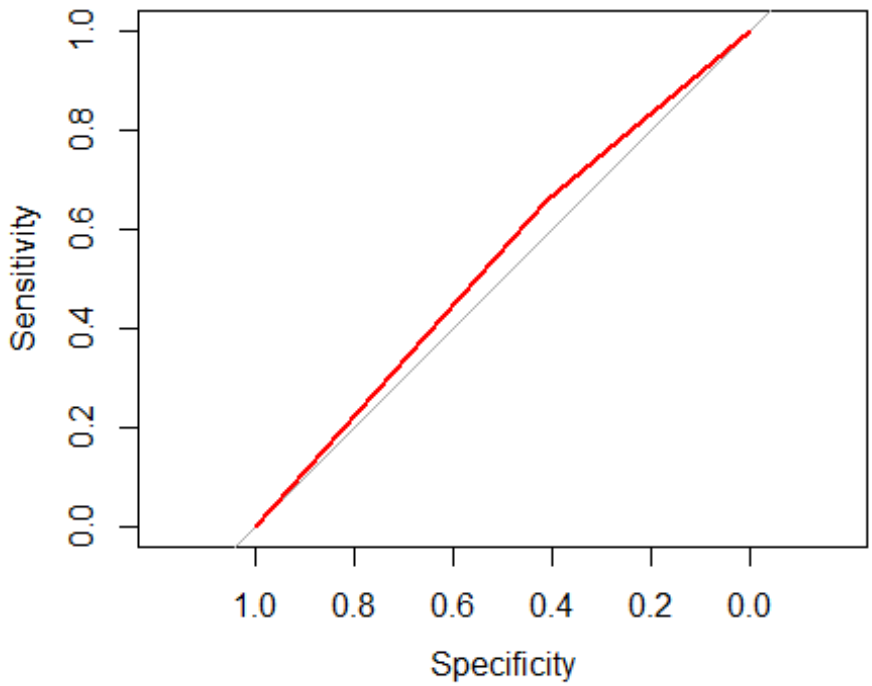
```
MatrixConfusionCJHW <- matrizconfusion(df$class, clustersH)
MatrixConfusionCJHW
```

```
##       Cluster
## Clase  1  2  3  4  5  6  7  8  9 10 11
##     1  0 17 12  0  0  0  0  0  0  0  0
##     2  0 72 38  0  0  0  0  0  0  0  0
##     3  0 38 22 30  0  0  0  0 14  0  0
##     4 31  0  0 44 19  0  0  0 21  0  0
##     5 45  0  0  0 57 14  0  0  0  0  0
##     6  1  0  0  0 27 66  4  0  0  0  0
##     7  0  0  0  0  0  9 45  1  0  0 43
##     8  0 37 10  0  0  0  2 49  0  0  5
##     9  0 31 20 31  0  0  0  1 12  0  0
##     10 0  0  0 25  0  0  0  0 14 79  0
##     11 0  0  0  0  0  0  0  0  0 14  0
```

```
#Calculamos la precision del modelo
PrecisionHW <- precision(MatrixConfusionCJHW)
PrecisionHW
```

```
## [1] 0.446
```

```
#Generamos la curva de ROC
modeloHW <- roc(df$class, clustersH)
plot(modeloHW,type="l",col="red")
```



```
##
## Call:
## roc.default(response = df$class, predictor = clustersH)
##
## Data: clustersH in 29 controls (df$class 1) < 110 cases (df$class 2).
## Area under the curve: 0.5342
```

Mejor modelo

```
#-------------------------------------------------------------------------
----------
#                                   MEJOR MODELO
#-------------------------------------------------------------------------
----------
precisiones <- c(PrecisionK, PrecisionDC, PrecisionHC, PrecisionDS,
PrecisionHS, PrecisionDA,
                 PrecisionHA, PrecisionDW, PrecisionHW)
x <- which.max(precisiones)
mejormodelo <- bestmodel(x)
cat("El mejor modelo es: ", mejormodelo, ", que posee una precision de: ",
precisiones[x], ".")

## El mejor modelo es:  CLASIFICACION JERARQUICA K: METHOD AVERAGE , que
posee una precision de:  0.457 .
```

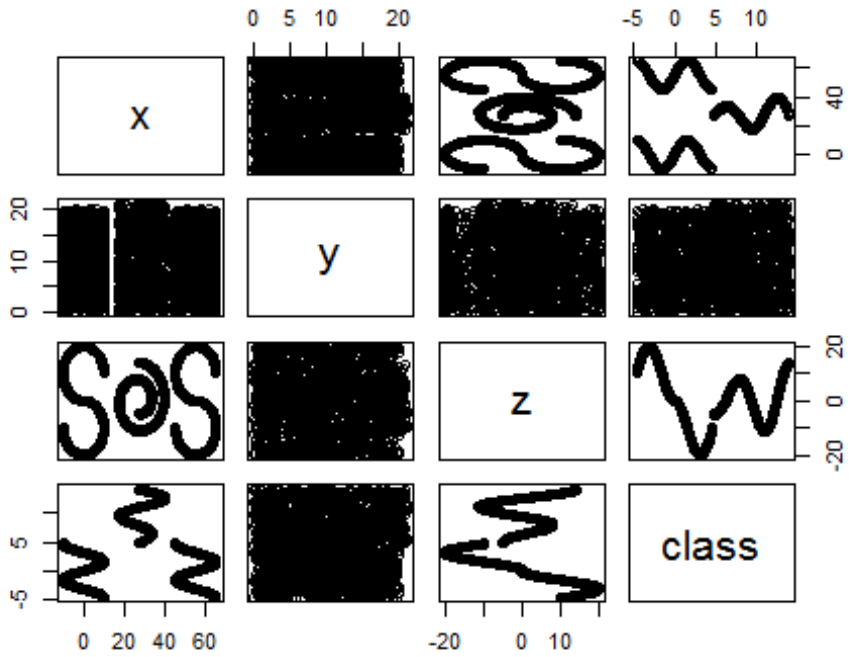
help.csv

- Preprocesamiento:

```
setwd("C:/Users/Eric/Desktop/AprendizajeNoSupervisado")
name = "help.csv"
#Lectura de datos.
df = read.csv(file =
"C:/Users/Eric/Desktop/AprendizajeNoSupervisado/data/help.csv", header = F)
#Modificamos el nombre de las columnas por comodidad.
colnames(df) <- c("x","y","z","class")
#Ordenamos la columna clase
df <- df[ order(df$class), ]
```

- Analisis exploratorio del dataset:

```
#*************************************************************************
************
#                    Analisis exploratorio del dataset
#*************************************************************************
************
#Matriz de dispersion
plot(df)
```




```
#. Cuántos clústers ve en el dataset help ?
#3

#. Qué pasa al aplicar la regla de asignación de clases en este dataset?
#Las 2 S las clasifican mal.

#. Qué solución daría para asignar de manera correcta los valores de las
clases y
#pueda analizar el desempeño del algoritmo de clustering de manera correcta?
#Generar las reglas por partes, utilice un criterio de filas pares e impares
para clasificar las clases de las S, y la O le asigne la clase faltante
```

Reglas para asignar las clases:

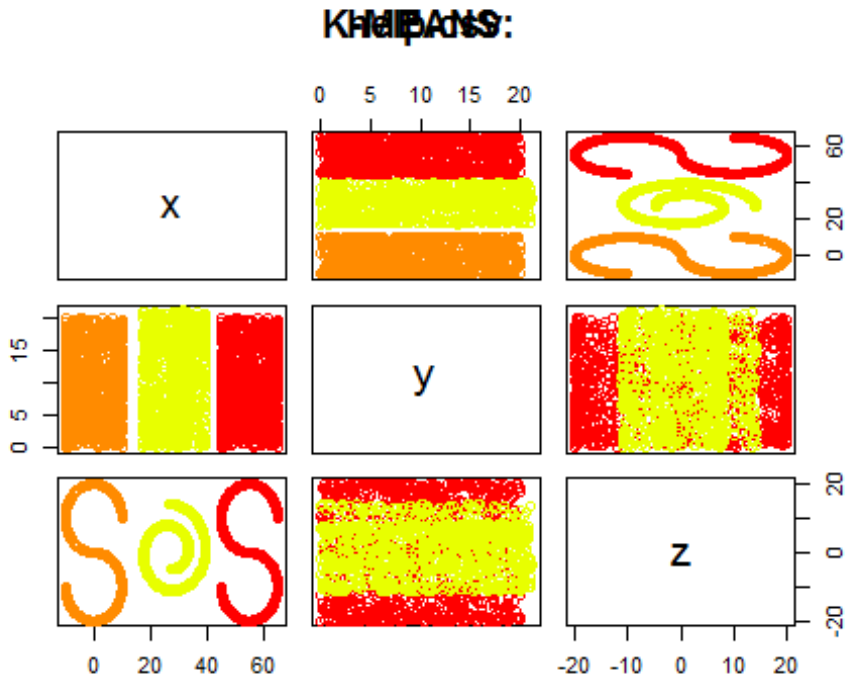
Ordene el dataframe por clases y utilice un criterio de filas pares e impares para clasificar las clases de las S, y la el otro conglomerado le asigne la clase faltante.

```
#**********************REGLA PARA ASIGNAR CLASES*****************************
reglas <- function(x,i){

  if ((i %% 2 == 0) == TRUE){
    return(1)
  }else{
    return(2)
  }
```

```
}

for (x in 1:2000) {

  df$class[x] <- reglas(df$class[x],x)
}

df$class[2001:3000] <- 3

plot3d(df$x, df$y, df$z, type = "s",size = 2, col = df$class)


#Observamos cuantos elementos hay de cada clase.
table(df$class)

##
##    1    2    3
## 1000 1000 1000

#1    2    3
#1000  1000 1000

length(unique(df$class))

## [1] 3

#Existen 3 clases.
```

K-medias:

```
#**************************************************************************
************
                                        #K-MEDIAS
#**************************************************************************
************
#Aplicamos k=3 ya que identificamos 3 conglomerados.
#kmedias(Dataframe, Columnas,K)
modeloK <- kmedias(df, 1:3, 3, name)
```



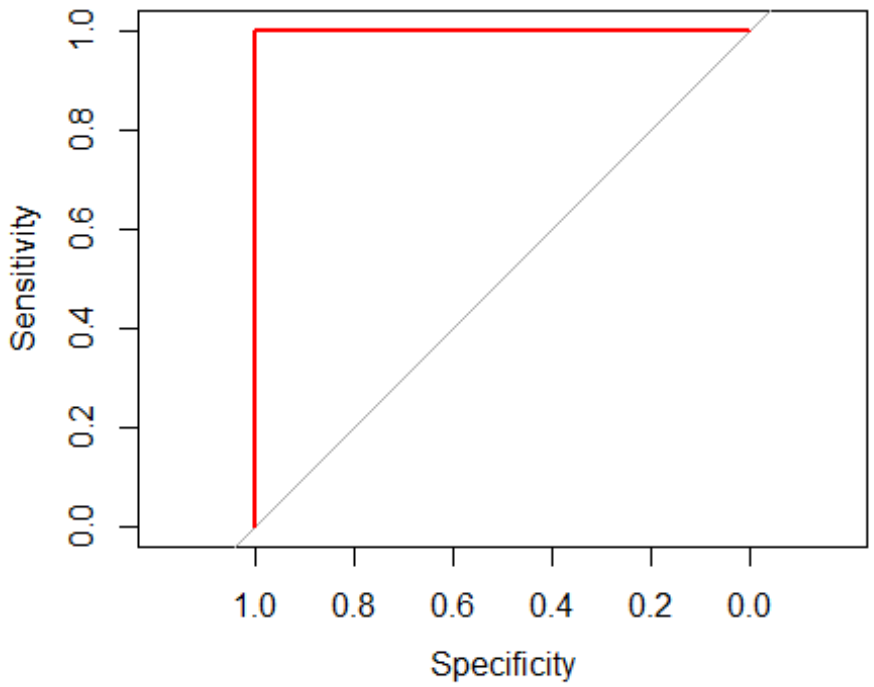

```
plot3d(df$x, df$y, df$z, type = "s",size = 2, col = modeloK$cluster)
#Generamos la matriz de confusion
MatrixConfusionK <- matrizconfusion(df$class,modeloK$cluster)
MatrixConfusionK
```

```
##       Cluster
## Clase    1    2    3
##     1 1000    0    0
##     2    0 1000    0
##     3    0    0 1000
```

```
#Calculamos la precision del modelo
PrecisionK <- precision(MatrixConfusionK)
PrecisionK
```

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

```
#Generamos la curva de ROC
modeloKROC <- roc(df$class, modeloK$cluster)
plot(modeloKROC,type="l",col="red")
```



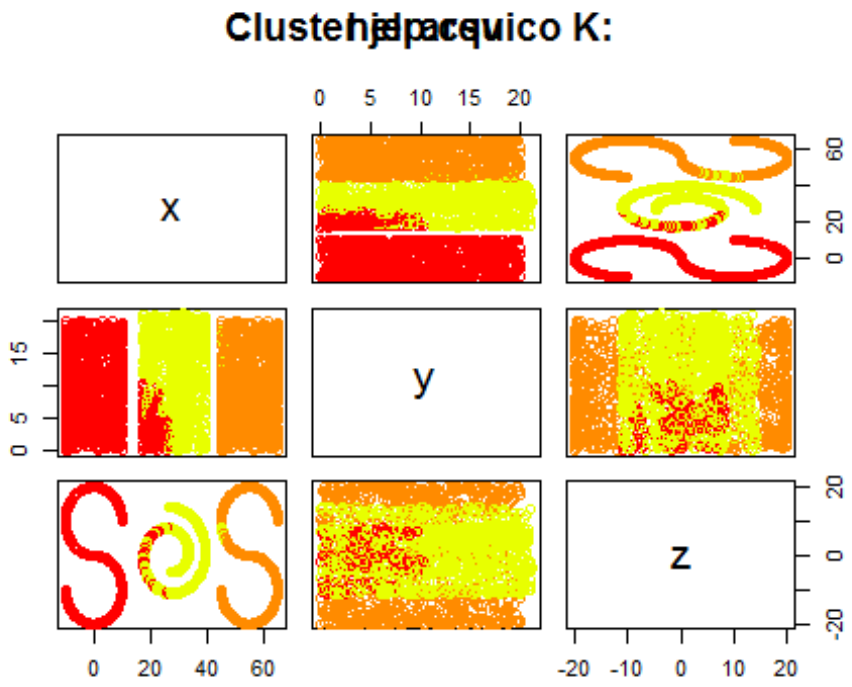

```
##
## Call:
## roc.default(response = df$class, predictor = modeloK$cluster)
##
## Data: modeloK$cluster in 1000 controls (df$class 1) < 1000 cases (df$class
2).
## Area under the curve: 1
```

Clusters jerárquicos:

```
#*************************************************************************
************
                                #Cluster Jerarquicos
#*************************************************************************
************
#Copy del dataset
datos = df
#Elimino la columna clase para realizar aprendizaje no supervisado.
datos$class <- NULL
#Convierto el dataframe en una matriz
datos= as.matrix(datos)
#Calculamos la matriz de distancia
distancia = dist(datos)
```

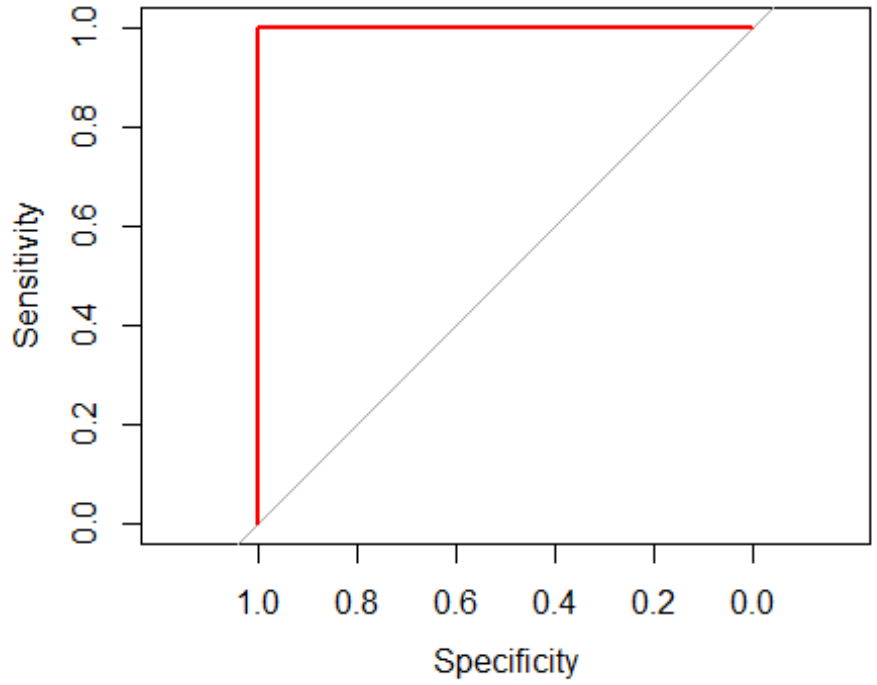
Método complete:

```
#-------------------------------------------------------------------------
----------
#                                  METHOD COMPLETE
#-------------------------------------------------------------------------
----------
#Dado un número de clústers k determinar la altura requerida
#para que tengamos el número de clúster k.
clustersD <- clusterJD(df, distancia, 1:3, "complete", 3, name)
```



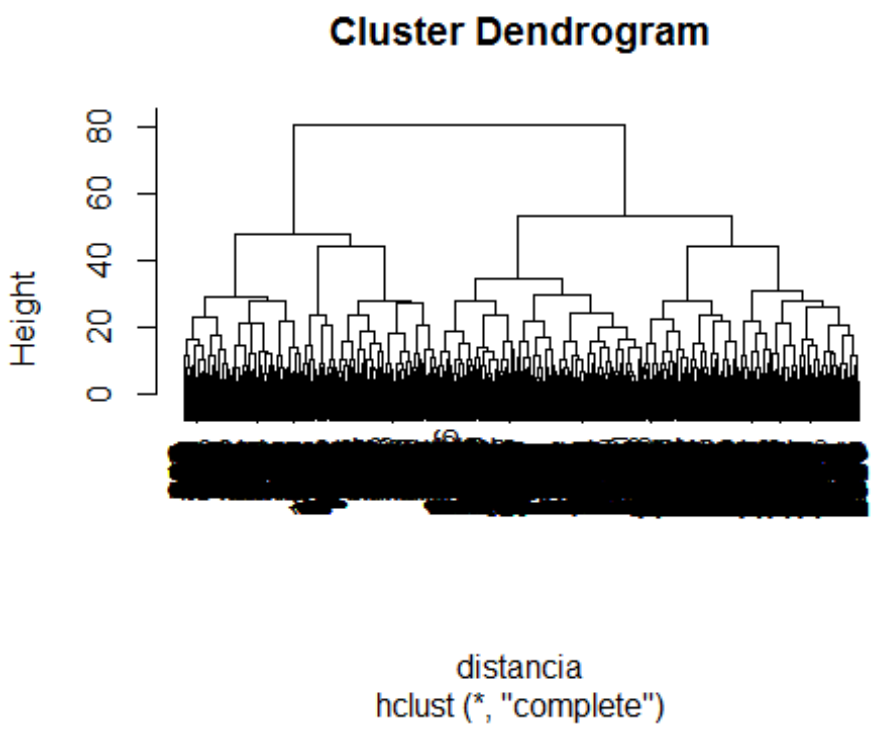

```
#Generamos la matriz de confusion
plot3d(df$x, df$y, df$z, type = "s",size = 2, col = clustersD)
MatrixConfusionCJDC <- matrizconfusion(df$class, clustersD)
MatrixConfusionCJDC

##      Cluster
## Clase    1    2    3
##     1  958    0   42
##     2    0 1000    0
##     3    0  132  868

#Calculamos la precision del modelo
PrecisionDC <- precision(MatrixConfusionCJDC)
PrecisionDC

## [1] 0.942
```

```r
#Generamos la curva de ROC
modeloDC <- roc(df$class, clustersD)
plot(modeloDC,type="l",col="red")
```

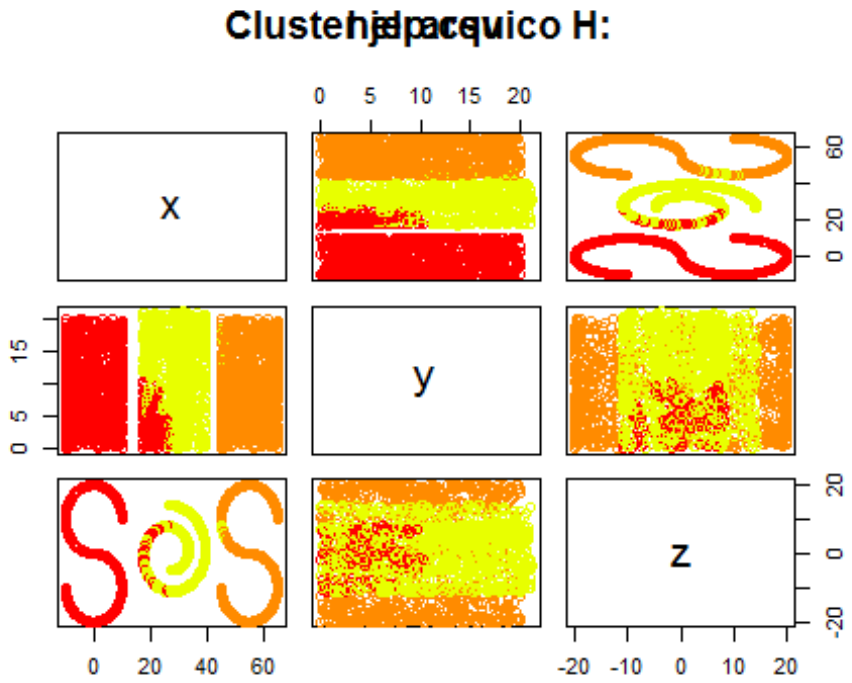



```
##
## Call:
## roc.default(response = df$class, predictor = clustersD)
##
## Data: clustersD in 1000 controls (df$class 1) > 1000 cases (df$class 2).
## Area under the curve: 1
```

```r
#**********************************************************
#Dada una altura h (una medida de disimilaridad) determinar
#el número de clústers que se obtienen.
clustersH <- clusterJH(df, distancia, 1:3, "complete", 50, name)
```




```
## [1] 1 2 3
```




```
plot3d(df$x, df$y, df$z, type = "s",size = 2, col = clustersH)
#Generamos la matriz de confusion
```

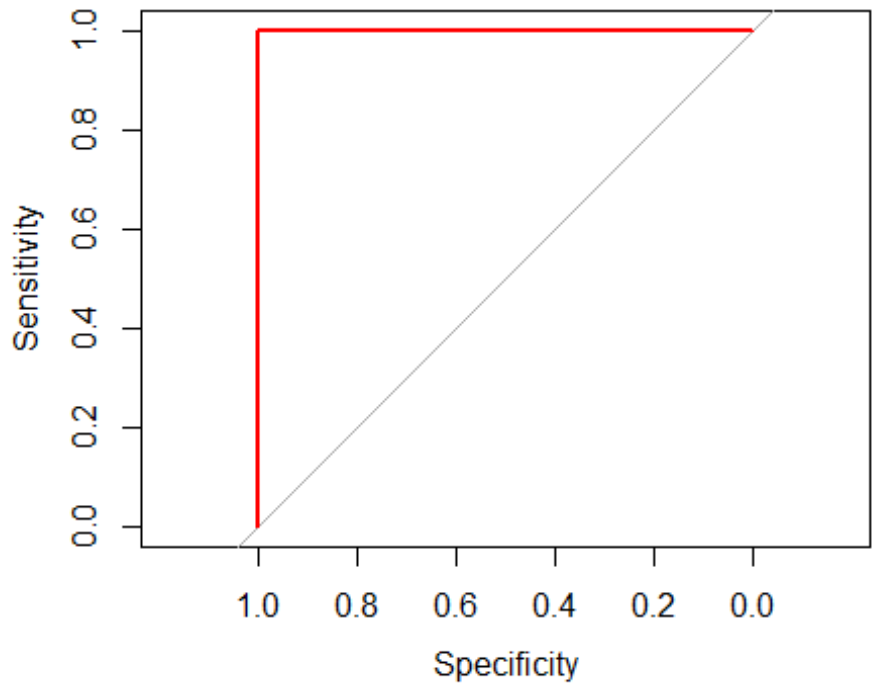
```r
MatrixConfusionCJHC <- matrizconfusion(df$class, clustersH)
MatrixConfusionCJHC
```

```
##      Cluster
## Clase    1    2    3
##     1  958    0   42
##     2    0 1000    0
##     3    0  132  868
```

```r
#Calculamos la precision del modelo
PrecisionHC <- precision(MatrixConfusionCJHC)
PrecisionHC
```

```
## [1] 0.942
```

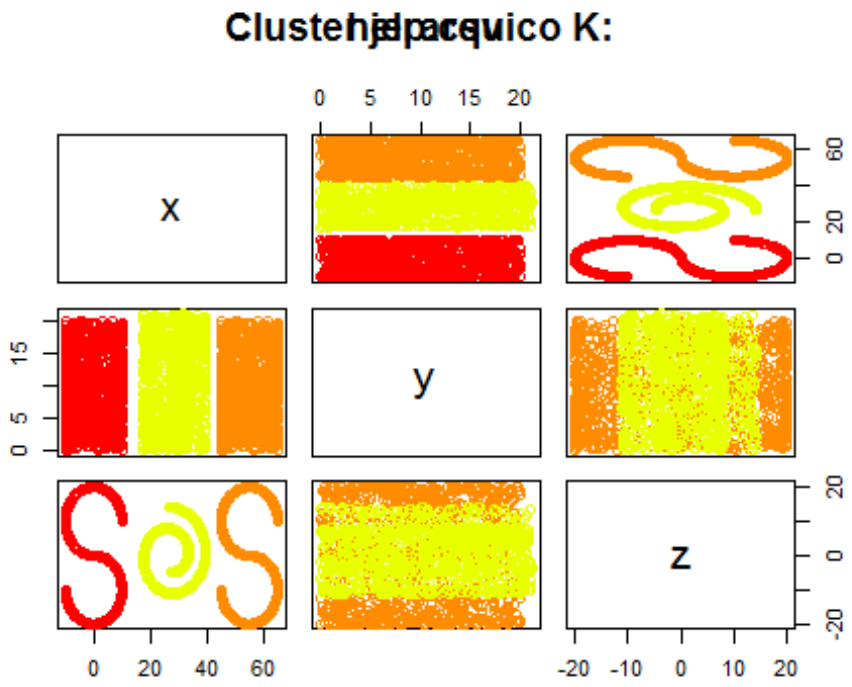
```r
#Generamos la curva de ROC
modeloHC <- roc(df$class, clustersH)
plot(modeloHC,type="l",col="red")
```




```
##
## Call:
## roc.default(response = df$class, predictor = clustersH)
##
## Data: clustersH in 1000 controls (df$class 1) > 1000 cases (df$class 2).
## Area under the curve: 1
```

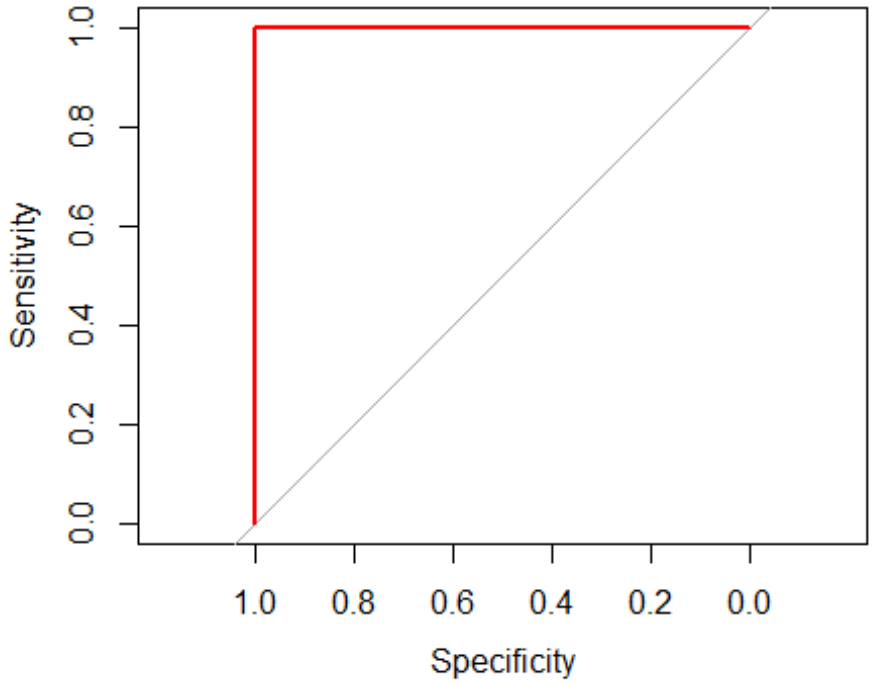
Método single:

```
#-------------------------------------------------------------------------
----------
#                          METHOD SINGLE
#-------------------------------------------------------------------------
----------
#Dado un número de clústers k determinar la altura requerida
#para que tengamos el número de clúster k.
clustersD <- clusterJD(df, distancia, 1:3, "single", 3, name)
```



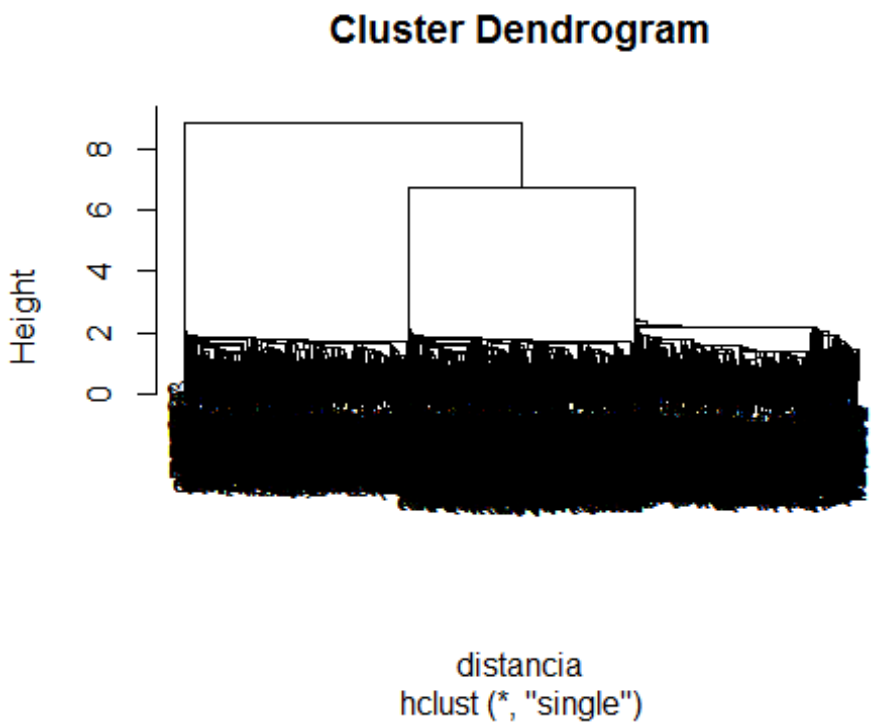

```
plot3d(df$x, df$y, df$z, type = "s",size = 2, col = clustersD)
#Generamos la matriz de confusion
MatrixConfusionCJDS <- matrizconfusion(df$class, clustersD)
MatrixConfusionCJDS

##      Cluster
## Clase    1    2    3
##     1 1000    0    0
##     2    0 1000    0
##     3    0    0 1000

#Calculamos la precision del modelo
PrecisionDS <- precision(MatrixConfusionCJDS)
PrecisionDS

## [1] 1
```

```r
#Generamos la curva de ROC
modeloDS <- roc(df$class, clustersD)
plot(modeloDS,type="l",col="red")
```

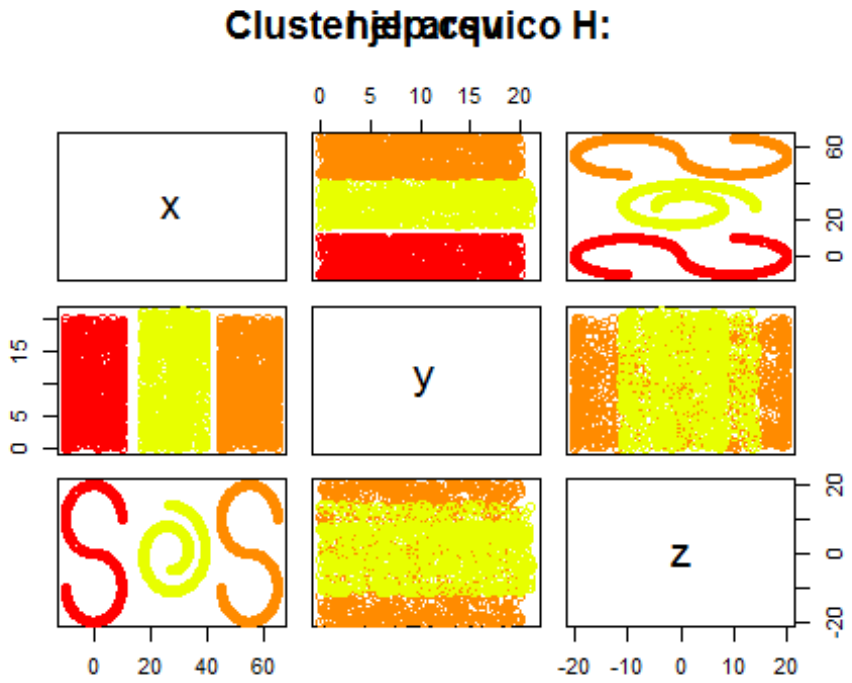



```
##
## Call:
## roc.default(response = df$class, predictor = clustersD)
##
## Data: clustersD in 1000 controls (df$class 1) > 1000 cases (df$class 2).
## Area under the curve: 1
```

```r
#**********************************************************
#Dada una altura h (una medida de disimilaridad) determinar
#el número de clústers que se obtienen.
clustersH <- clusterJH(df, distancia, 1:3, "single", 5, name)
```




```
## [1] 1 2 3
```




```
plot3d(df$x, df$y, df$z, type = "s",size = 2, col = clustersH)
#Generamos la matriz de confusion
```

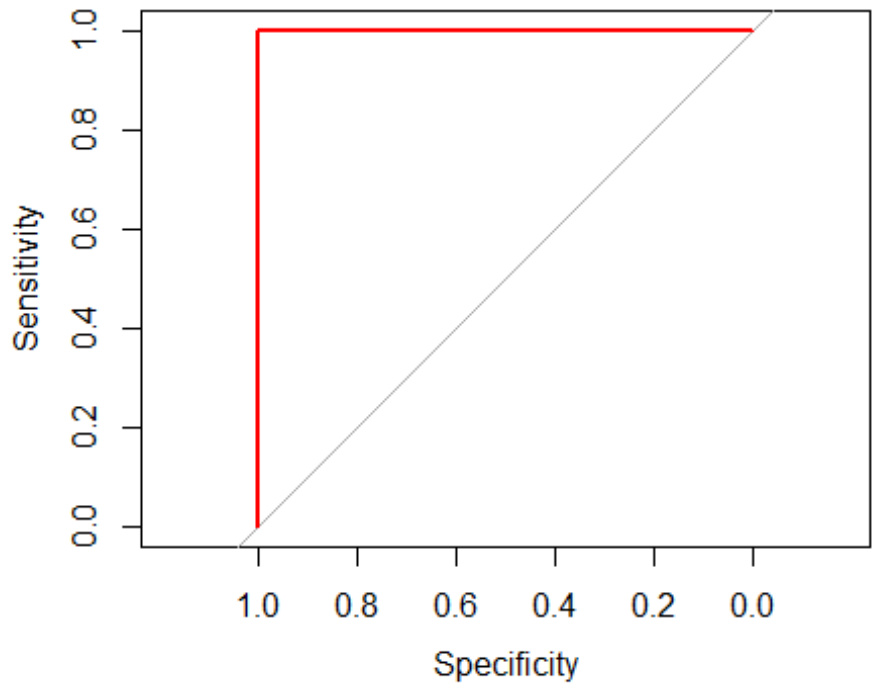
```
MatrixConfusionCJHS <- matrizconfusion(df$class, clustersH)
MatrixConfusionCJHS
```

```
##      Cluster
## Clase    1    2    3
##     1 1000    0    0
##     2    0 1000    0
##     3    0    0 1000
```

```
#Calculamos la precision del modelo
PrecisionHS <- precision(MatrixConfusionCJHS)
PrecisionHS
```

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

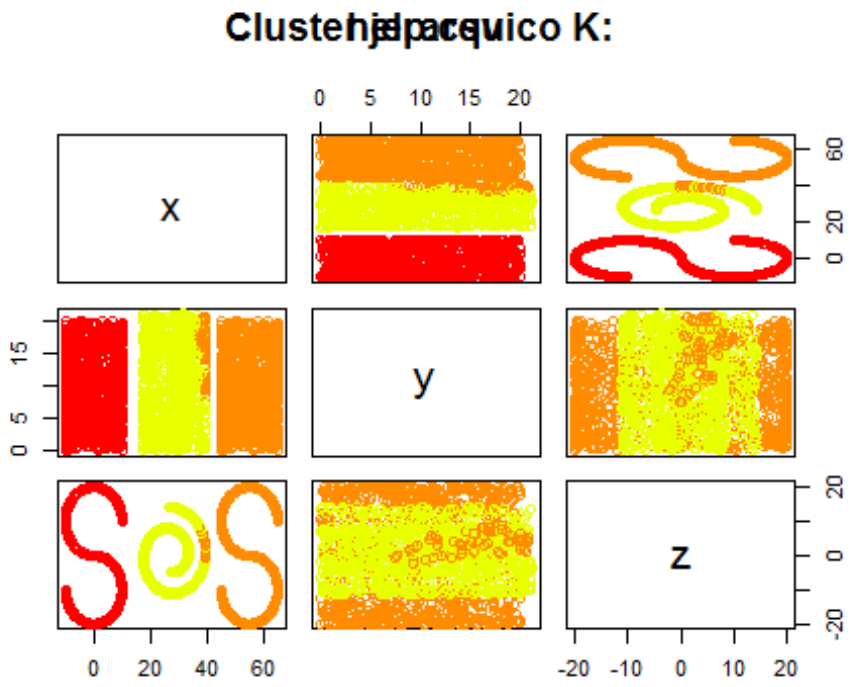
```
#Generamos la curva de ROC
modeloHS <- roc(df$class, clustersH)
plot(modeloHS,type="l",col="red")
```




```
##
## Call:
## roc.default(response = df$class, predictor = clustersH)
##
## Data: clustersH in 1000 controls (df$class 1) > 1000 cases (df$class 2).
## Area under the curve: 1
```

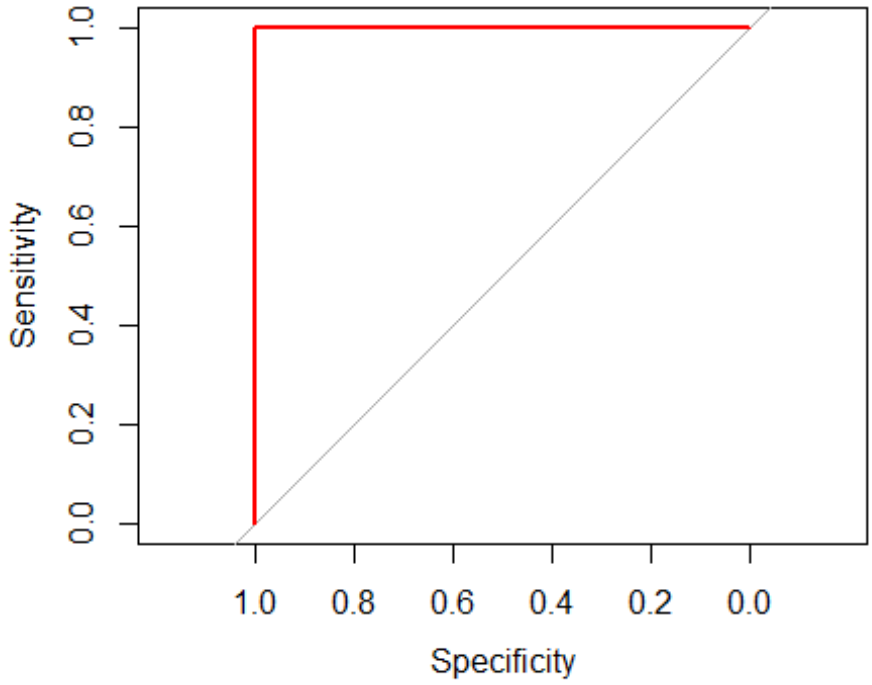
Método average:

```
#-------------------------------------------------------------------------
----------
#                              METHOD AVERAGE
#-------------------------------------------------------------------------
----------
#Dado un número de clústers k determinar la altura requerida
#para que tengamos el número de clúster k.
clustersD <- clusterJD(df, distancia, 1:3, "average", 3, name)
```



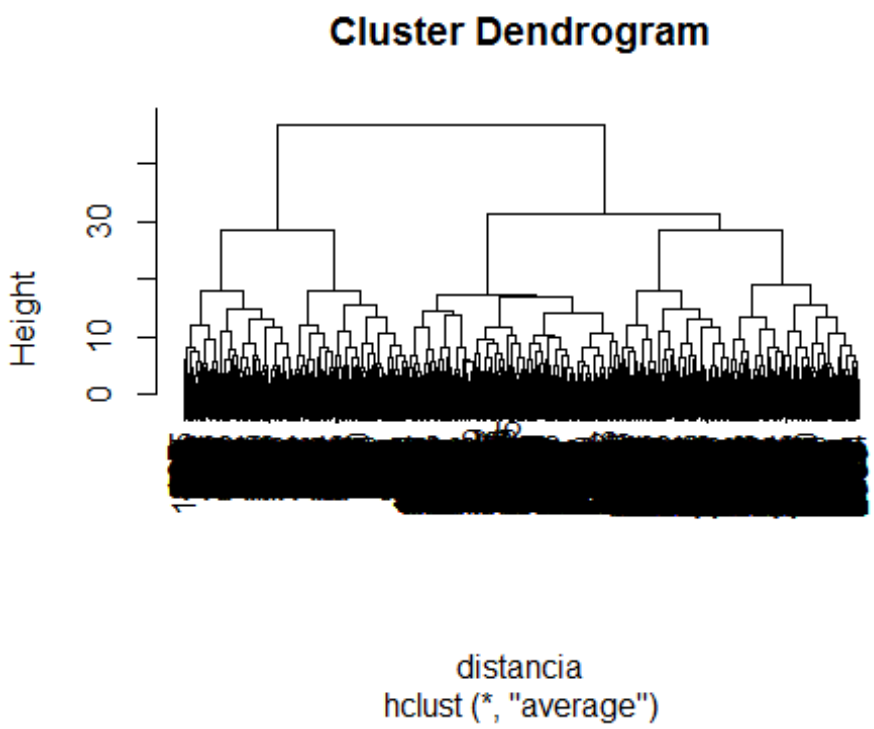

```
plot3d(df$x, df$y, df$z, type = "s",size = 2, col = clustersD)
#Generamos la matriz de confusion
MatrixConfusionCJDA <- matrizconfusion(df$class, clustersD)
MatrixConfusionCJDA

##      Cluster
## Clase    1    2    3
##     1 1000    0    0
##     2    0 1000    0
##     3   56    0  944

#Calculamos la precision del modelo
PrecisionDA <- precision(MatrixConfusionCJDA)
PrecisionDA

## [1] 0.9813333
```

```r
#Generamos la curva de ROC
modeloDA <- roc(df$class, clustersD)
plot(modeloDA,type="l",col="red")
```

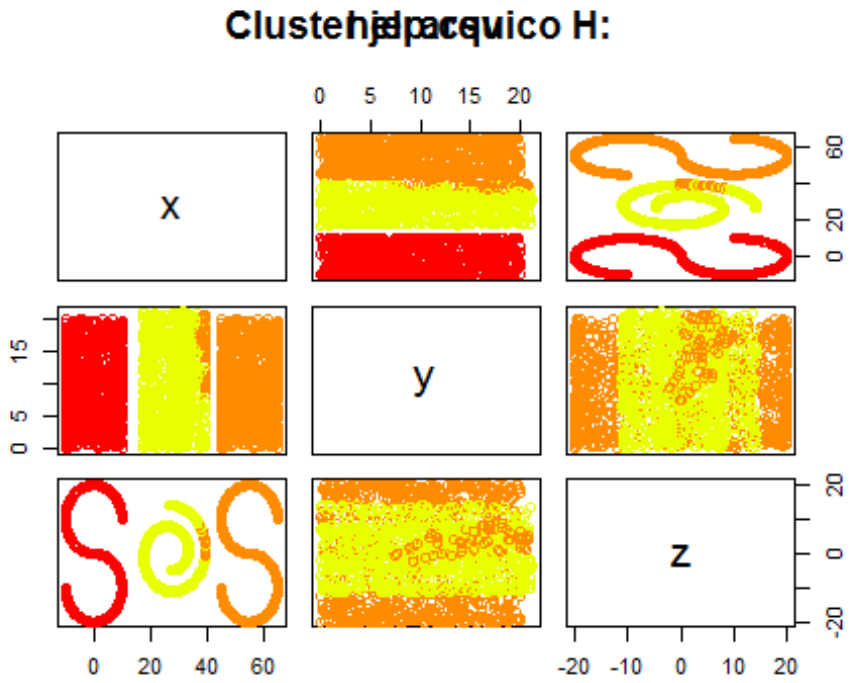



```
##
## Call:
## roc.default(response = df$class, predictor = clustersD)
##
## Data: clustersD in 1000 controls (df$class 1) > 1000 cases (df$class 2).
## Area under the curve: 1
```

```r
#**********************************************************
#Dada una altura h (una medida de disimilaridad) determinar
#el número de clústers que se obtienen.
clustersH <- clusterJH(df, distancia, 1:3, "average", 30, name)
```




```
## [1] 1 2 3
```




```
plot3d(df$x, df$y, df$z, type = "s",size = 2, col = clustersH)
#Generamos la matriz de confusion
```

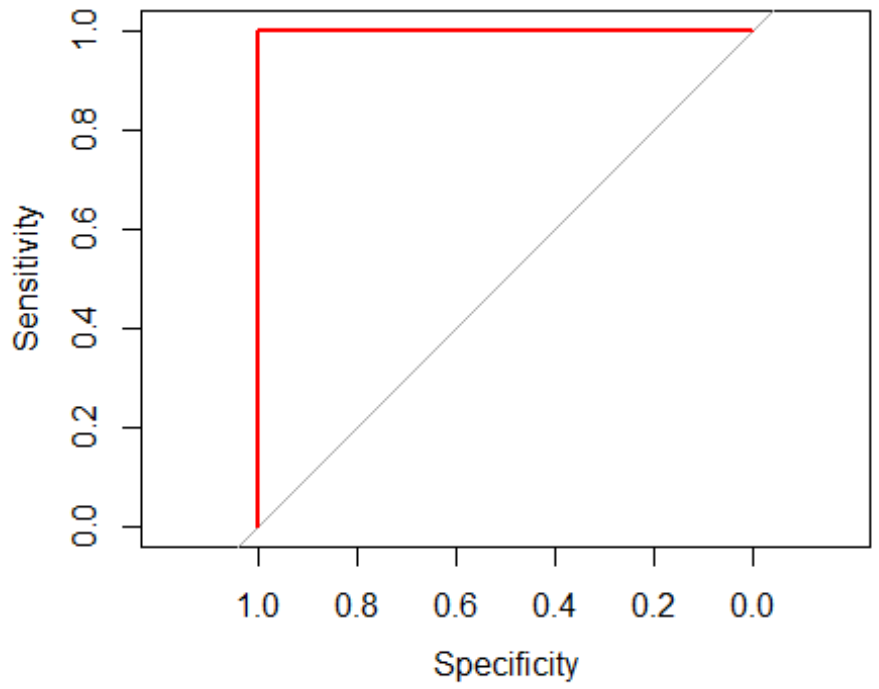
```
MatrixConfusionCJHA <- matrizconfusion(df$class, clustersH)
MatrixConfusionCJHA
```

```
##      Cluster
## Clase    1    2    3
##     1 1000    0    0
##     2    0 1000    0
##     3   56    0  944
```

```
#Calculamos la precision del modelo
PrecisionHA <- precision(MatrixConfusionCJHA)
PrecisionHA
```

```
## [1] 0.9813333
```

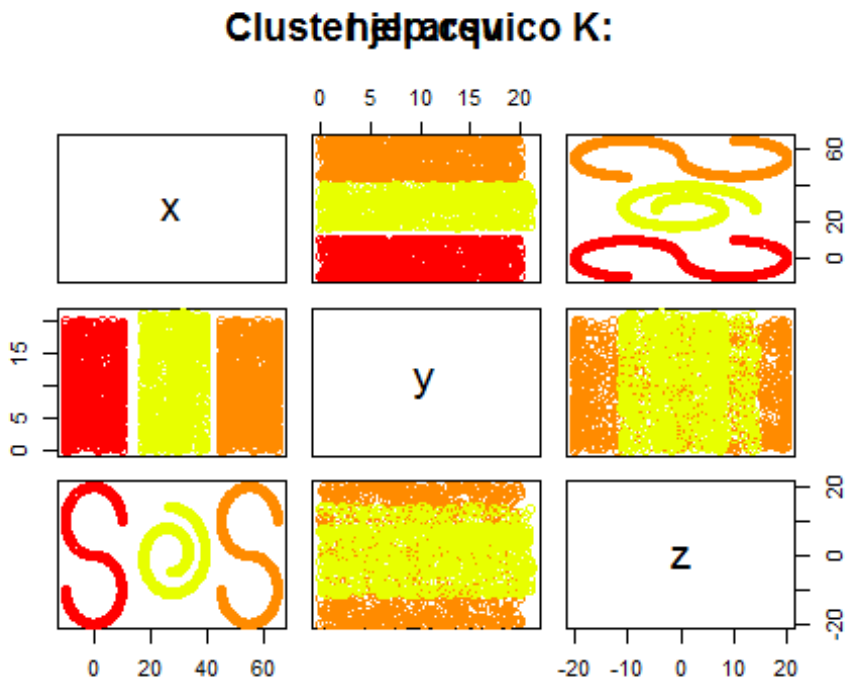
```
#Generamos la curva de ROC
modeloHA <- roc(df$class, clustersH)
plot(modeloHA,type="l",col="red")
```




```
##
## Call:
## roc.default(response = df$class, predictor = clustersH)
##
## Data: clustersH in 1000 controls (df$class 1) > 1000 cases (df$class 2).
## Area under the curve: 1
```

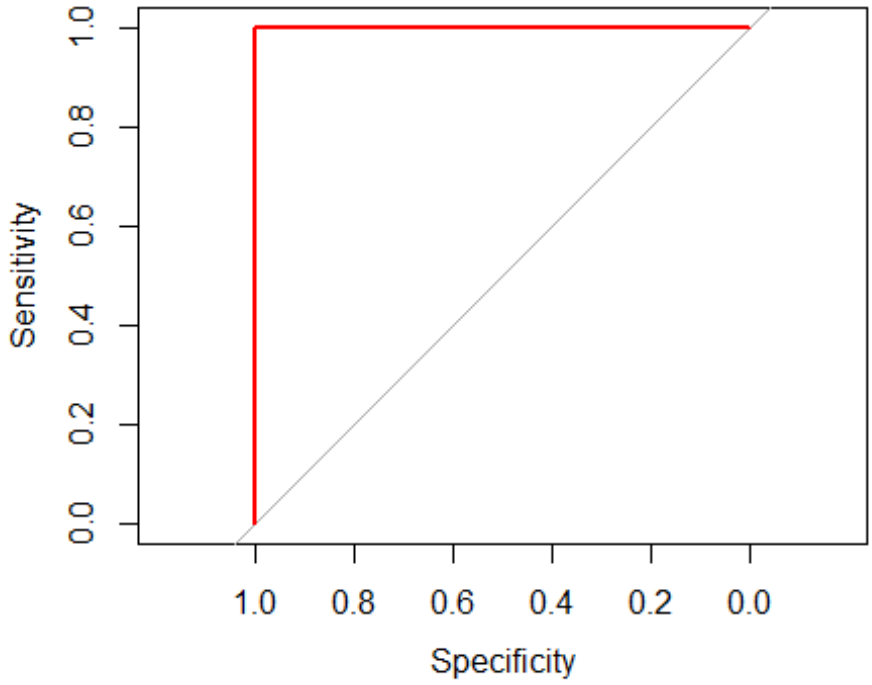
Método ward.D

```
#-------------------------------------------------------------------------
----------
#                                METHOD ward.D
#-------------------------------------------------------------------------
----------
#Dado un número de clústers k determinar la altura requerida
#para que tengamos el número de clúster k.
clustersD <- clusterJD(df, distancia, 1:3, "ward.D", 3, name)
```



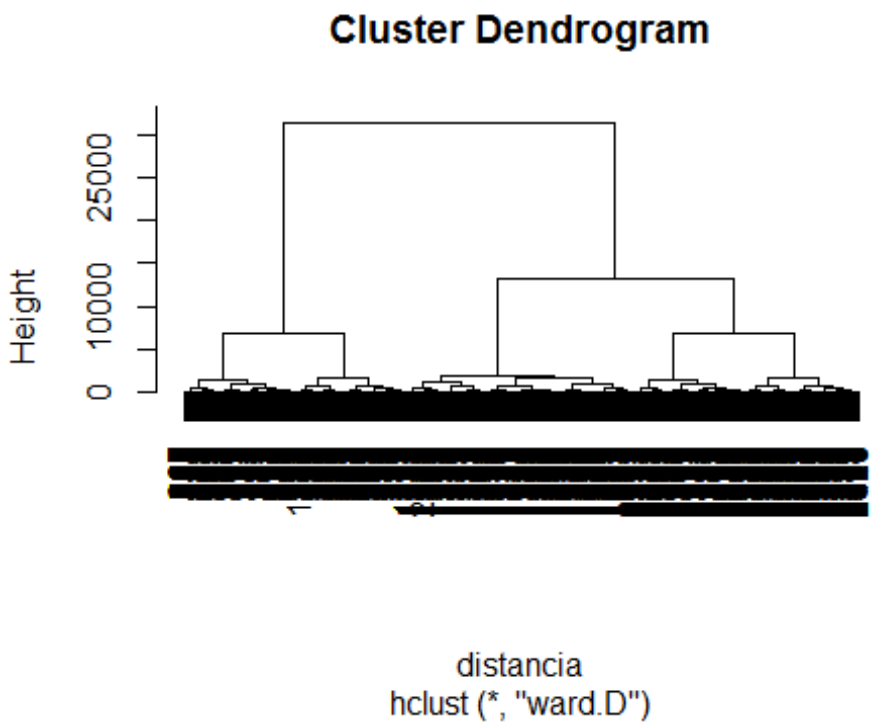

```
plot3d(df$x, df$y, df$z, type = "s",size = 2, col = clustersD)
#Generamos la matriz de confusion
MatrixConfusionCJDW <- matrizconfusion(df$class, clustersD)
MatrixConfusionCJDW

##      Cluster
## Clase    1    2    3
##     1 1000    0    0
##     2    0 1000    0
##     3    0    0 1000

#Calculamos la precision del modelo
PrecisionDW <- precision(MatrixConfusionCJDW)
PrecisionDW

## [1] 1
```

```r
#Generamos la curva de ROC
modeloDW <- roc(df$class, clustersD)
plot(modeloDW,type="l",col="red")
```

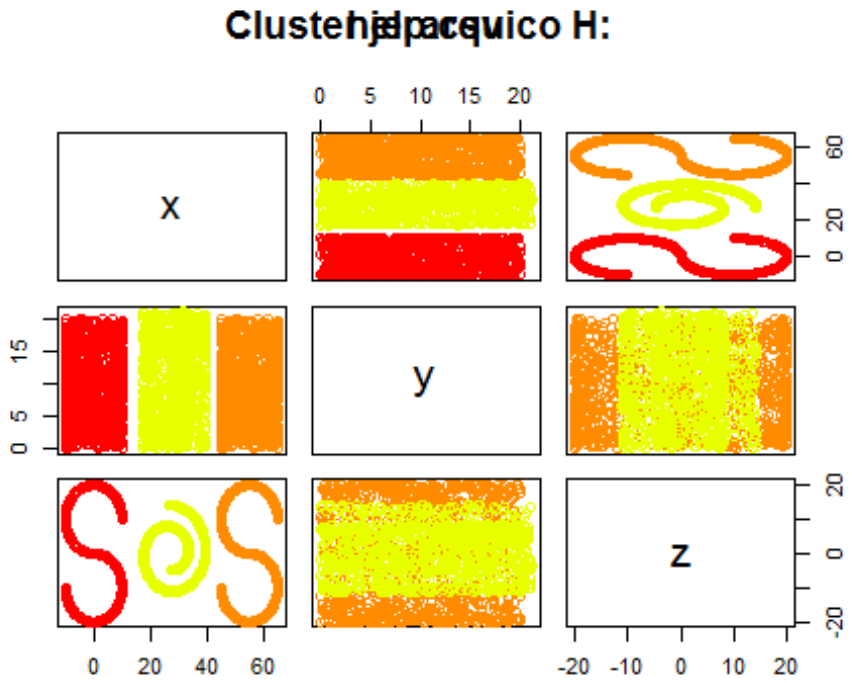



```
##
## Call:
## roc.default(response = df$class, predictor = clustersD)
##
## Data: clustersD in 1000 controls (df$class 1) > 1000 cases (df$class 2).
## Area under the curve: 1
```

```r
#*********************************************************
#Dada una altura h (una medida de disimilaridad) determinar
#el número de clústers que se obtienen.
clustersH <- clusterJH(df, distancia, 1:3, "ward.D", 10000, name)
```




```
## [1] 1 2 3
```




```
plot3d(df$x, df$y, df$z, type = "s",size = 2, col = clustersH)
#Generamos la matriz de confusion
```

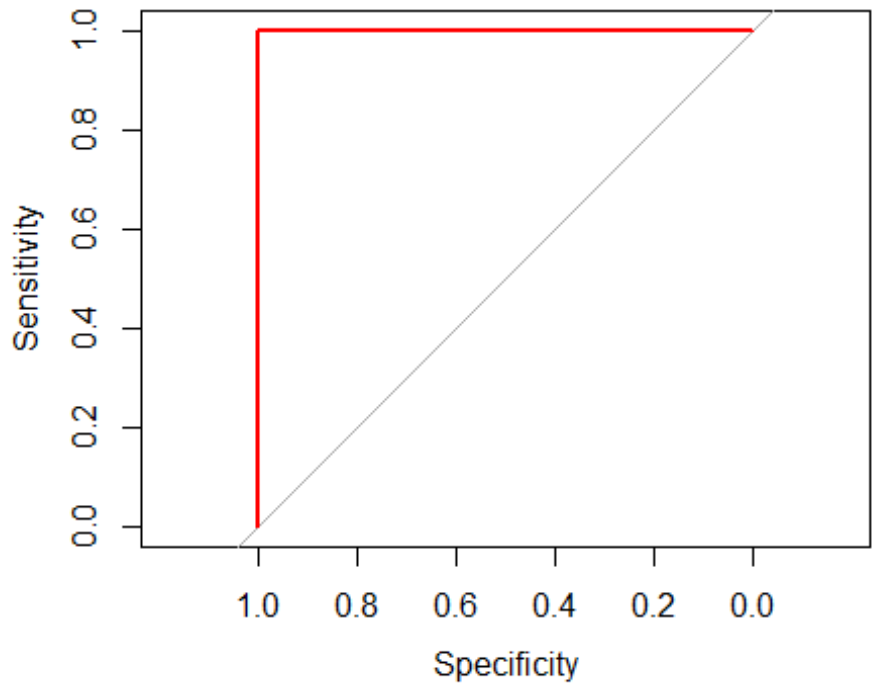
```
MatrixConfusionCJHW <- matrizconfusion(df$class, clustersH)
MatrixConfusionCJHW
```

```
##      Cluster
## Clase    1    2    3
##     1 1000    0    0
##     2    0 1000    0
##     3    0    0 1000
```

```
#Calculamos la precision del modelo
PrecisionHW <- precision(MatrixConfusionCJHW)
PrecisionHW
```

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

```
#Generamos la curva de ROC
modeloHW <- roc(df$class, clustersH)
plot(modeloHW,type="l",col="red")
```




```
##
## Call:
## roc.default(response = df$class, predictor = clustersH)
##
## Data: clustersH in 1000 controls (df$class 1) > 1000 cases (df$class 2).
## Area under the curve: 1
```

Mejor modelo

```
#-------------------------------------------------------------------------
----------
#                                  MEJOR MODELO
#-------------------------------------------------------------------------
----------
precisiones <- c(PrecisionK, PrecisionDC, PrecisionHC, PrecisionDS,
PrecisionHS, PrecisionDA,
                 PrecisionHA, PrecisionDW, PrecisionHW)
x <- which.max(precisiones)
mejormodelo <- bestmodel(x)
cat("El mejor modelo es: ", mejormodelo, ", que posee una precision de: ",
precisiones[x], ".")

## El mejor modelo es:  K-MEDIAS , que posee una precision de:  1 .
```

moon.csv

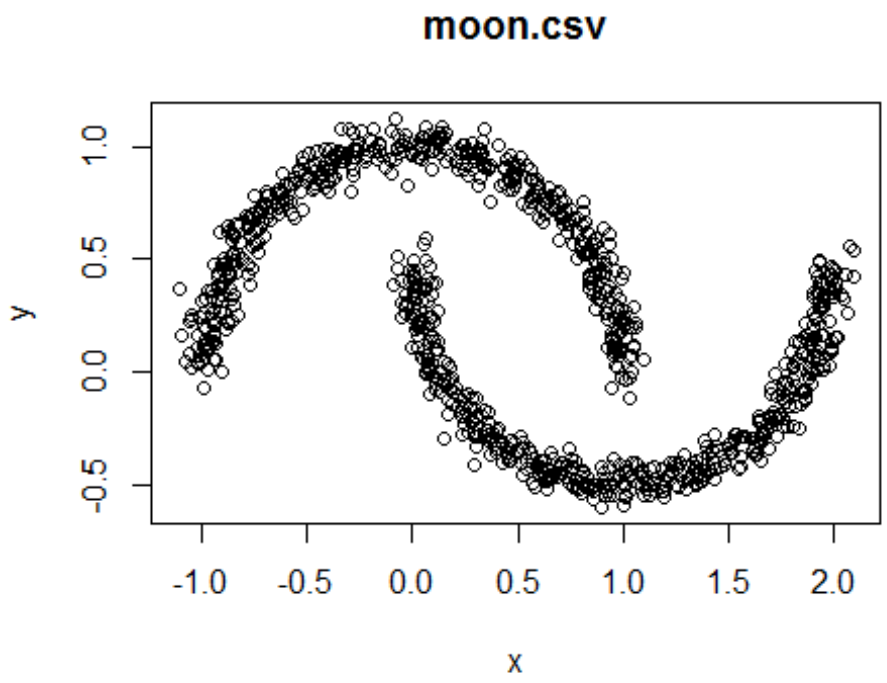
- Preprocesamiento:

```
setwd("C:/Users/Eric/Desktop/AprendizajeNoSupervisado")
name = "moon.csv"
#Lectura de datos.
df = read.csv(file =
"C:/Users/Eric/Desktop/AprendizajeNoSupervisado/data/moon.csv", header = F)
#Modificamos el nombre de las columnas por comodidad.
colnames(df) <- c("x","y","class")
df$class = as.numeric(df$class)
if (min(df$class) == 0){
  df$class <- df$class + 1
}
```

- Analisis exploratorio del dataset:

```
#*************************************************************************
************
#                      Analisis exploratorio del dataset
#*************************************************************************
************
#Podemos observar que hay 3 columnas.
head(df)

##           x         y class
## 1  0.438270  0.900774     1
## 2 -0.767655  0.672376     1
## 3  0.487233  0.827758     1
## 4  0.904183 -0.446244     2
## 5  0.190063 -0.142372     2
## 6  0.113667  0.097442     2

#Observamos cuantos elementos hay de cada clase.
table(df$class)
```

```
##
##   1   2
## 500 500

#1   2
#500 500

#Grafico
plot(df$x, df$y, xlab = "x", ylab = "y", main = name)
```

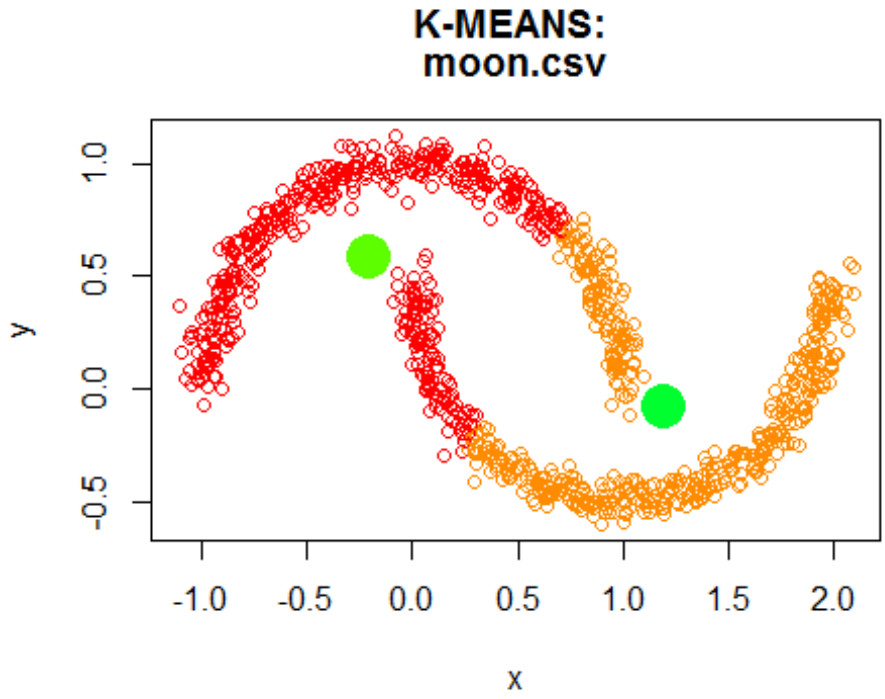



```
#Podemos observar 2 conglomerados

length(unique(df$class))

## [1] 2

#Existen 2 clases.
```

K-medias:

```
#**************************************************************************
************
#                       K-MEDIAS
#**************************************************************************
************
#Aplicamos k=2 ya que identificamos 2 conglomerados y existen 2 clases.
```

```r
#kmedias(Dataframe, Columnas,K)
modeloK <- kmedias(df, 1:2, 2, name)
```



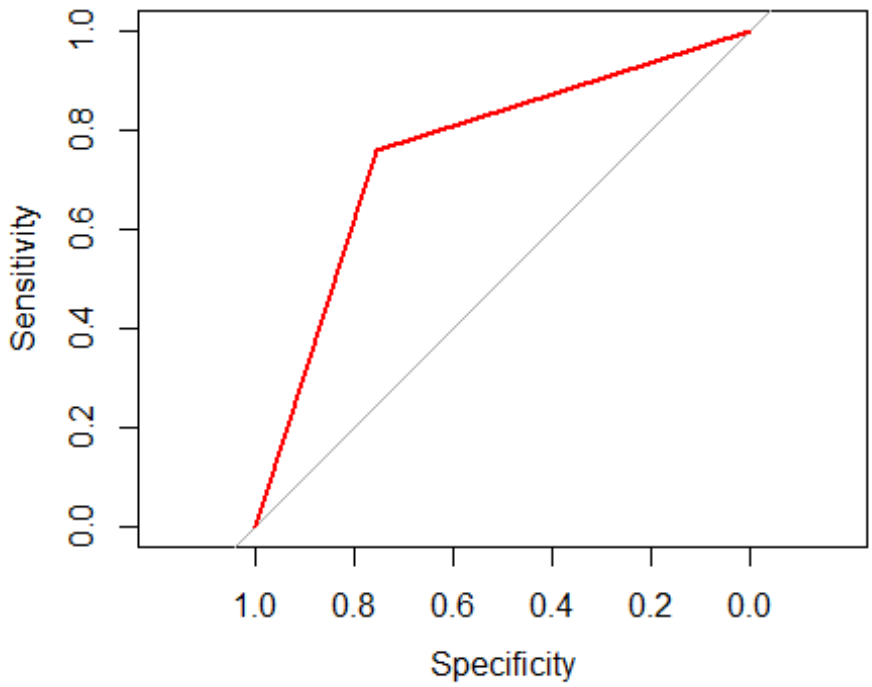

```r
#Generamos la matriz de confusion
MatrixConfusionK <- matrizconfusion(df$class,modeloK$cluster)
MatrixConfusionK
```

```
##      Cluster
## Clase   1   2
##     1 377 123
##     2 121 379
```

```r
#Calculamos la precision del modelo
PrecisionK <- precision(MatrixConfusionK)
PrecisionK
```

```
## [1] 0.756
```

```r
#Generamos la curva de ROC
modeloKROC <- roc(df$class, modeloK$cluster)
plot(modeloKROC,type="l",col="red")
```



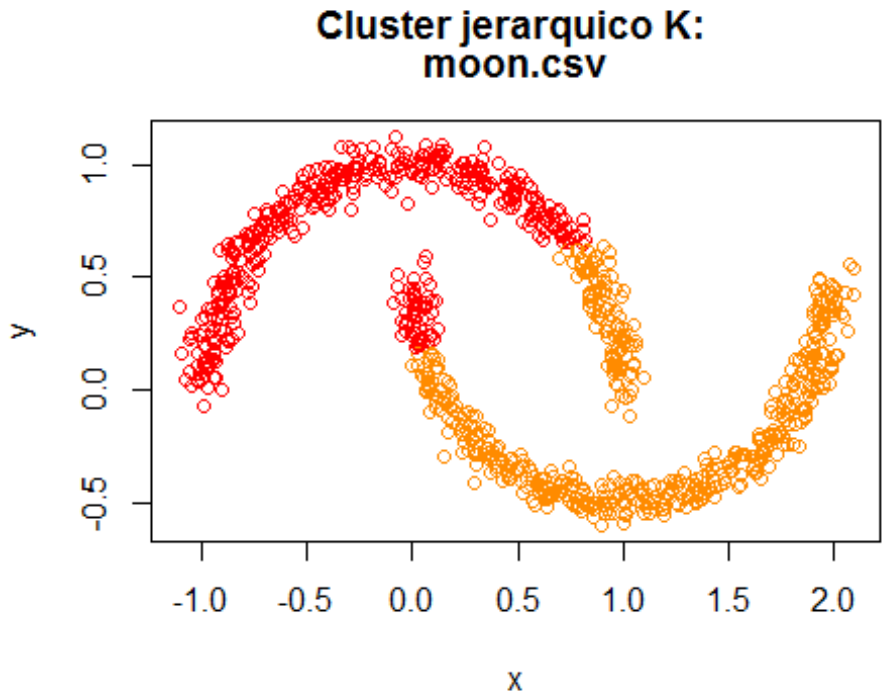

```
##
## Call:
## roc.default(response = df$class, predictor = modeloK$cluster)
##
## Data: modeloK$cluster in 500 controls (df$class 1) < 500 cases (df$class
2).
## Area under the curve: 0.756
```

Clusters jerárquicos:

```
#*************************************************************************
************
                                #Cluster Jerárquicos
#*************************************************************************
************
#Copy del dataset
datos = df
#Elimino la columna clase para realizar aprendizaje no supervisado.
datos$class <- NULL
#Convierto el dataframe en una matriz
datos= as.matrix(datos)
#Calculamos la matriz de distancia
distancia = dist(datos)
```

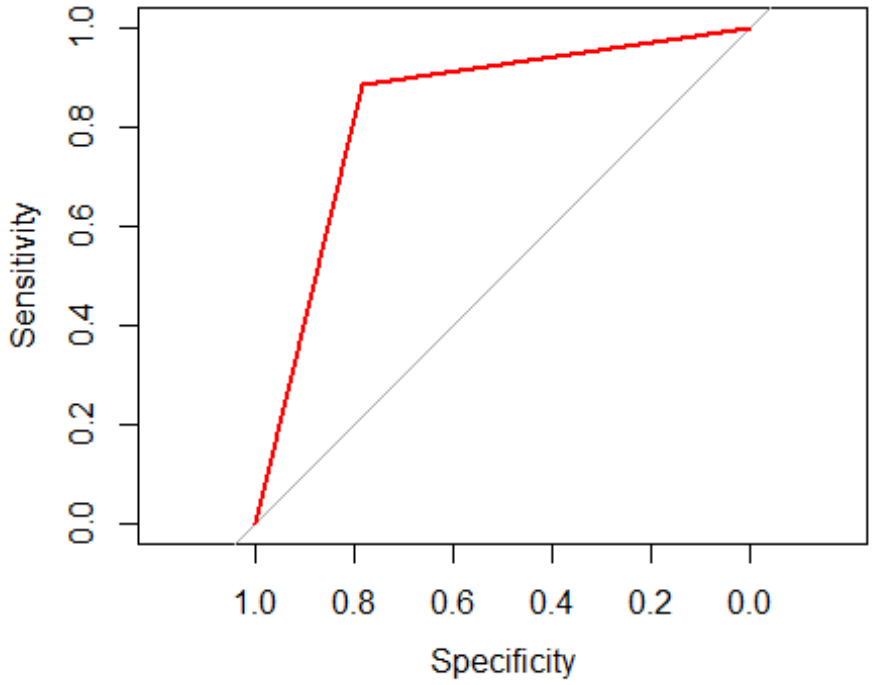
Método complete:

```
#-------------------------------------------------------------------------
----------
#                                METHOD COMPLETE
#-------------------------------------------------------------------------
----------
#Dado un número de clústers k determinar la altura requerida
#para que tengamos el número de clúster k.
clustersD <- clusterJD(df, distancia, 1:2, "complete", 2, name)
```



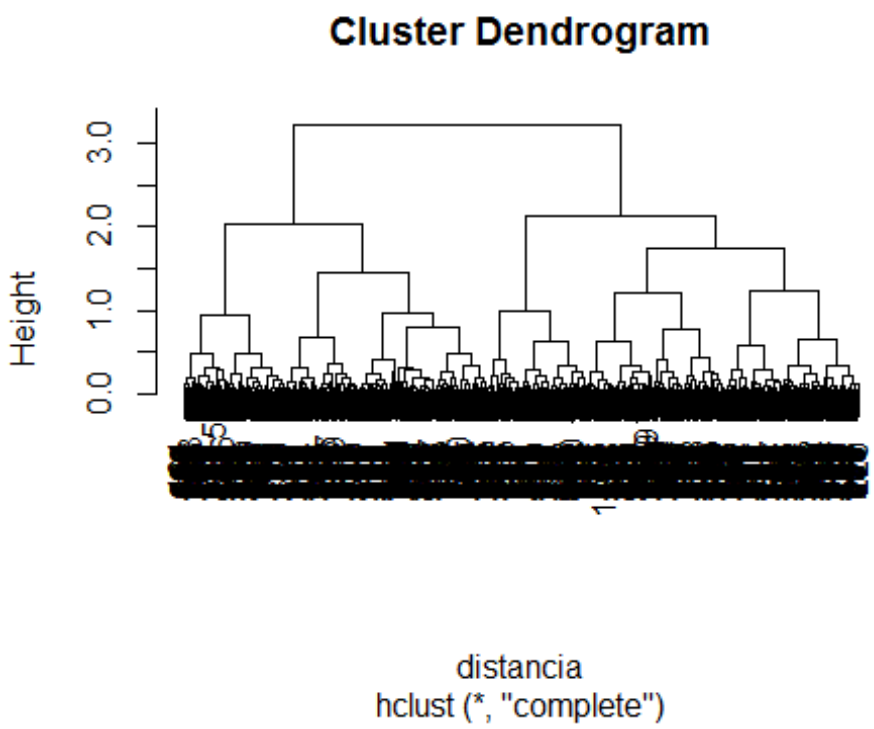

```
#Generamos la matriz de confusion
MatrixConfusionCJDC <- matrizconfusion(df$class, clustersD)
MatrixConfusionCJDC

##      Cluster
## Clase   1   2
##     1 393 107
##     2  57 443

#Calculamos la precision del modelo
PrecisionDC <- precision(MatrixConfusionCJDC)
PrecisionDC

## [1] 0.836
```

```r
#Generamos la curva de ROC
modeloDC <- roc(df$class, clustersD)
plot(modeloDC,type="l",col="red")
```

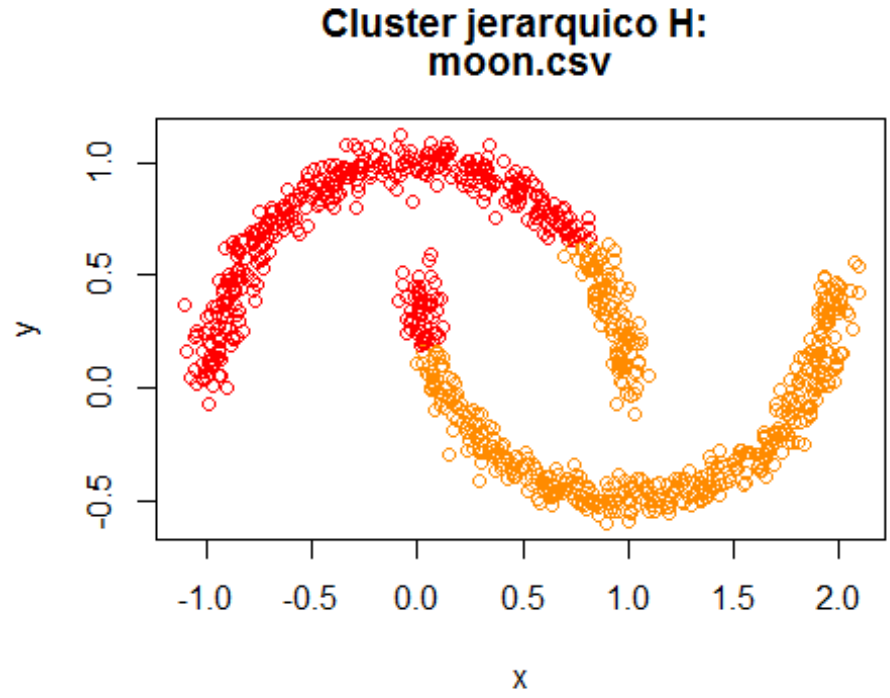



```
##
## Call:
## roc.default(response = df$class, predictor = clustersD)
##
## Data: clustersD in 500 controls (df$class 1) < 500 cases (df$class 2).
## Area under the curve: 0.836
```

```r
#*********************************************************
#Dada una altura h (una medida de disimilaridad) determinar
#el número de clústers que se obtienen.
clustersH <- clusterJH(df, distancia, 1:2, "complete", 3, name)
```


Cluster Dendrogram
Height
0.0
1.0
2.0
3.0
distancia
hclust (*, "complete")

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


Cluster jerarquico H:
moon.csv
y
-0.5
0.0
0.5
1.0
x
-1.0
-0.5
0.0
0.5
1.0
1.5
2.0

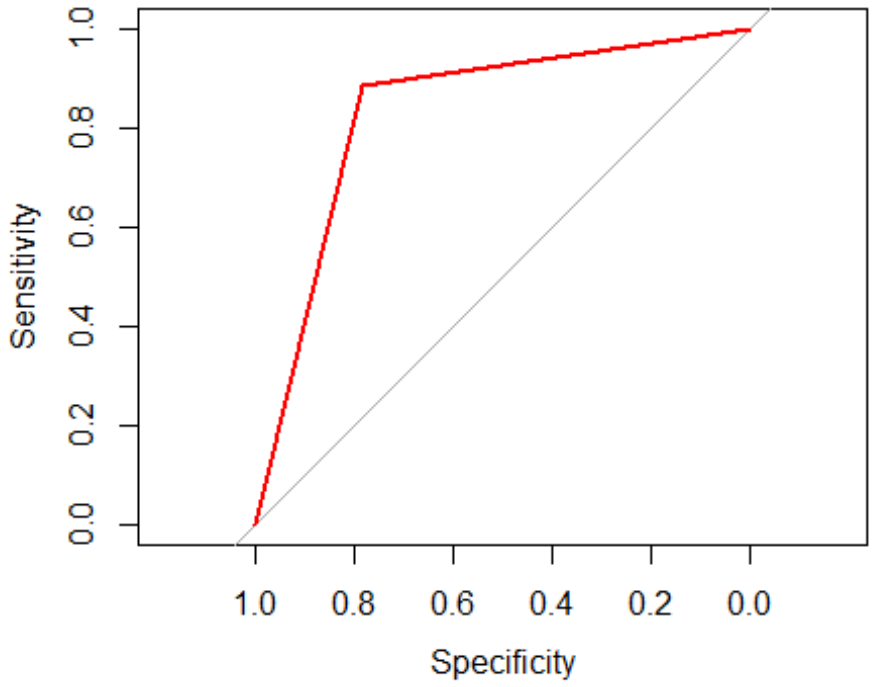
```r
#Generamos la matriz de confusion
MatrixConfusionCJHC <- matrizconfusion(df$class, clustersH)
MatrixConfusionCJHC
```

```
##      Cluster
## Clase   1   2
##     1 393 107
##     2  57 443
```

```r
#Calculamos la precision del modelo
PrecisionHC <- precision(MatrixConfusionCJHC)
PrecisionHC
```

```
## [1] 0.836
```

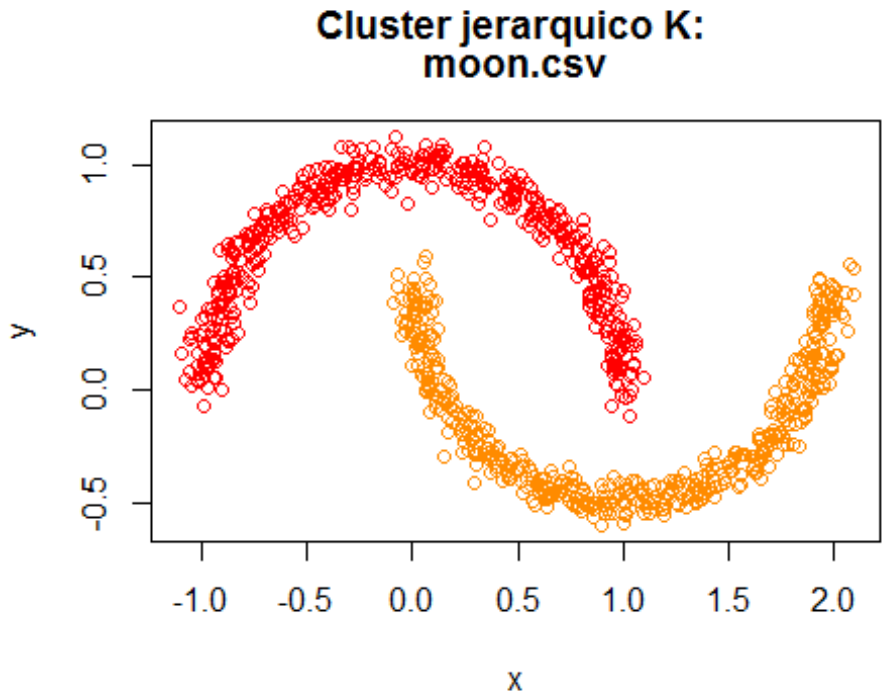
```r
#Generamos la curva de ROC
modeloHC <- roc(df$class, clustersH)
plot(modeloHC,type="l",col="red")
```




```
##
## Call:
## roc.default(response = df$class, predictor = clustersH)
##
## Data: clustersH in 500 controls (df$class 1) < 500 cases (df$class 2).
## Area under the curve: 0.836
```

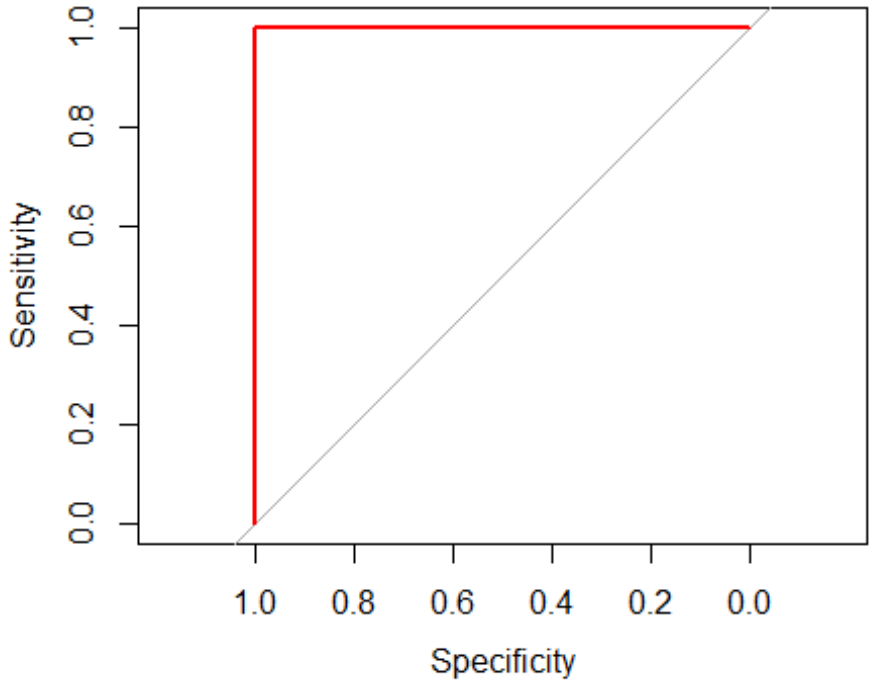
Método single:

```
#-------------------------------------------------------------------------
----------
#                          METHOD SINGLE
#-------------------------------------------------------------------------
----------
#Dado un número de clústers k determinar la altura requerida
#para que tengamos el número de clúster k.
clustersD <- clusterJD(df, distancia, 1:2, "single", 2, name)
```



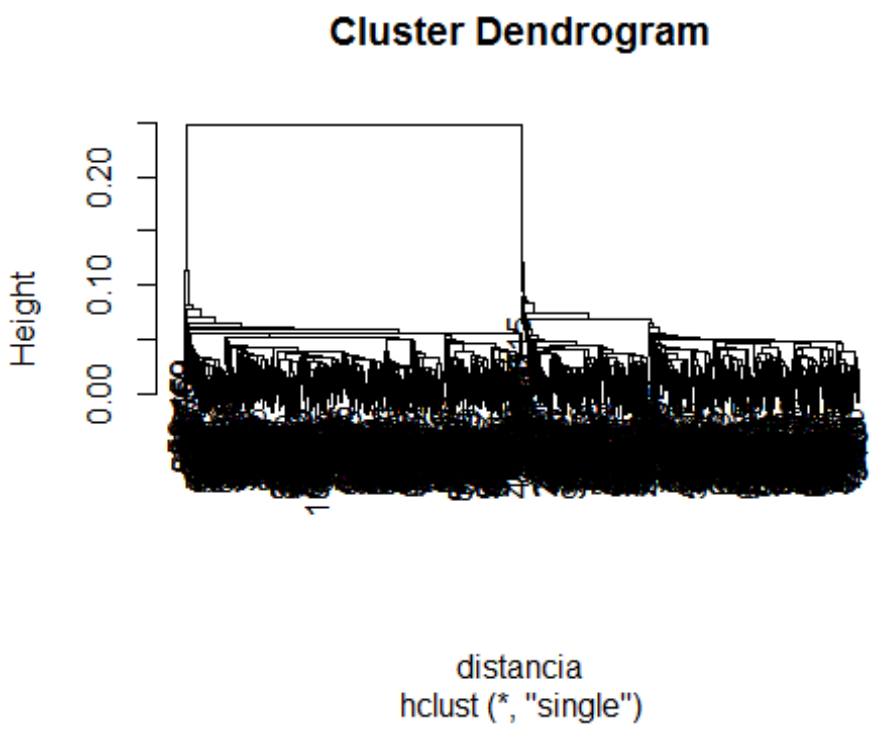

```
#Generamos la matriz de confusion
MatrixConfusionCJDS <- matrizconfusion(df$class, clustersD)
MatrixConfusionCJDS

##      Cluster
## Clase   1   2
##     1 500   0
##     2   0 500

#Calculamos la precision del modelo
PrecisionDS <- precision(MatrixConfusionCJDS)
PrecisionDS

## [1] 1
```

```r
#Generamos la curva de ROC
modeloDS <- roc(df$class, clustersD)
plot(modeloDS,type="l",col="red")
```

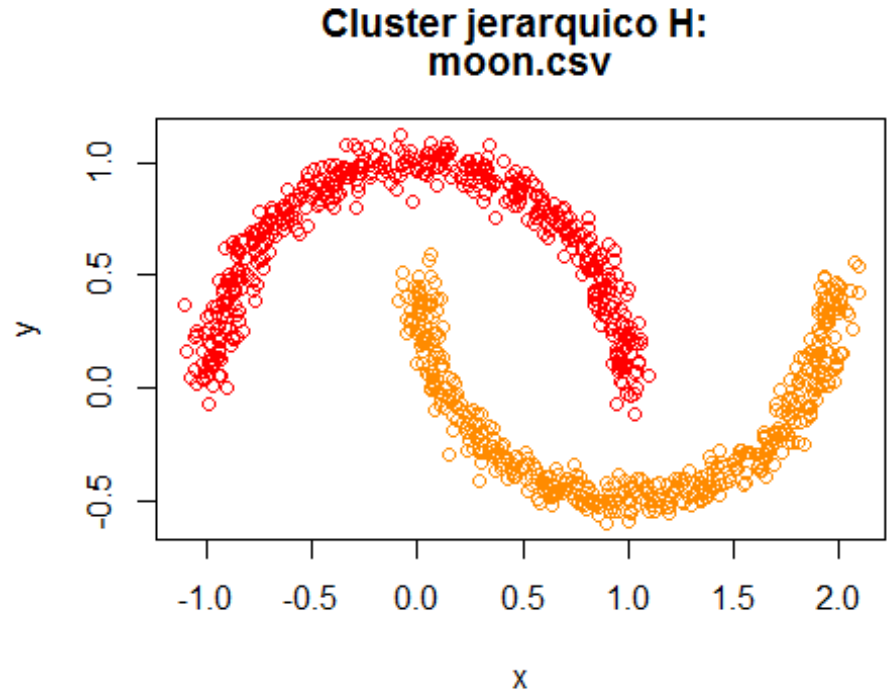



```
##
## Call:
## roc.default(response = df$class, predictor = clustersD)
##
## Data: clustersD in 500 controls (df$class 1) < 500 cases (df$class 2).
## Area under the curve: 1
```

```r
#*********************************************************
#Dada una altura h (una medida de disimilaridad) determinar
#el número de clústers que se obtienen.
clustersH <- clusterJH(df, distancia, 1:2, "single", 0.20, name)
```


Cluster Dendrogram
Height
0.00
0.10
0.20
distancia
hclust (*, "single")

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


Cluster jerarquico H:
moon.csv
y
1.0
0.5
0.0
-0.5
-1.0
-0.5
0.0
0.5
1.0
1.5
2.0
x

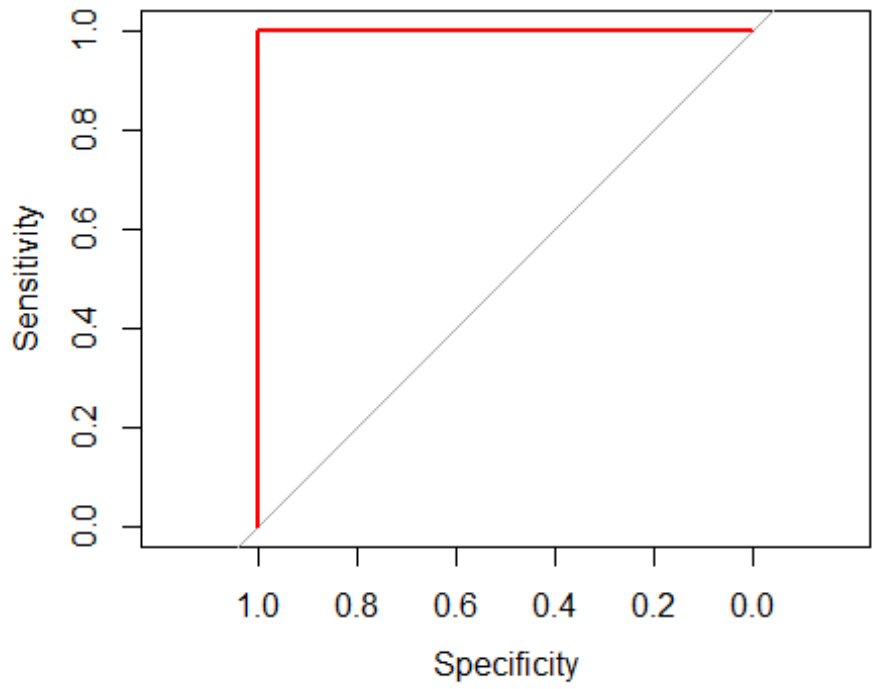
```r
#Generamos la matriz de confusion
MatrixConfusionCJHS <- matrizconfusion(df$class, clustersH)
MatrixConfusionCJHS
```

```
##      Cluster
## Clase   1   2
##     1 500   0
##     2   0 500
```

```r
#Calculamos la precision del modelo
PrecisionHS <- precision(MatrixConfusionCJHS)
PrecisionHS
```

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

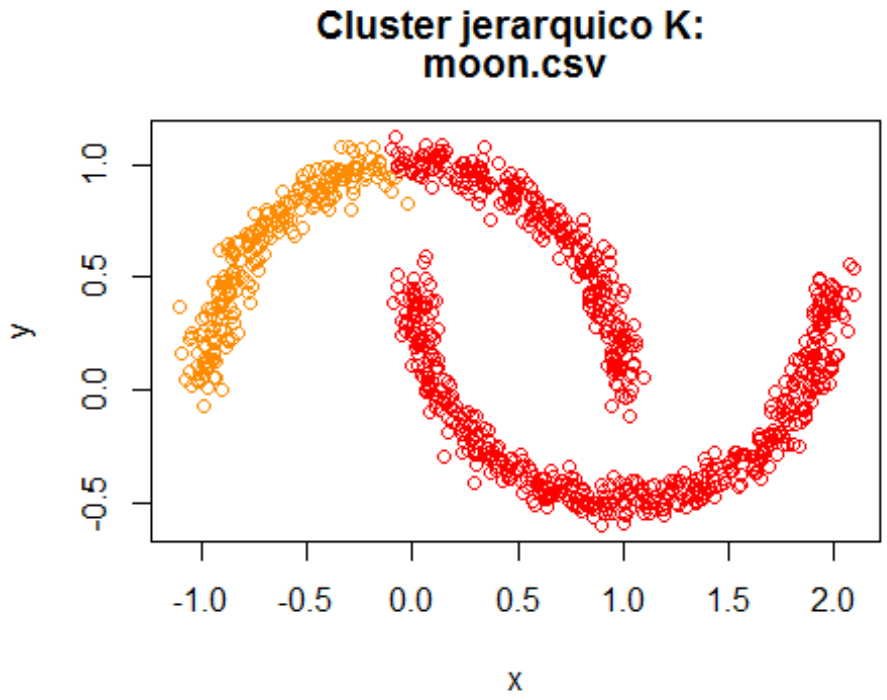
```r
#Generamos la curva de ROC
modeloHS <- roc(df$class, clustersH)
plot(modeloHS,type="l",col="red")
```




```
##
## Call:
## roc.default(response = df$class, predictor = clustersH)
##
## Data: clustersH in 500 controls (df$class 1) < 500 cases (df$class 2).
## Area under the curve: 1
```

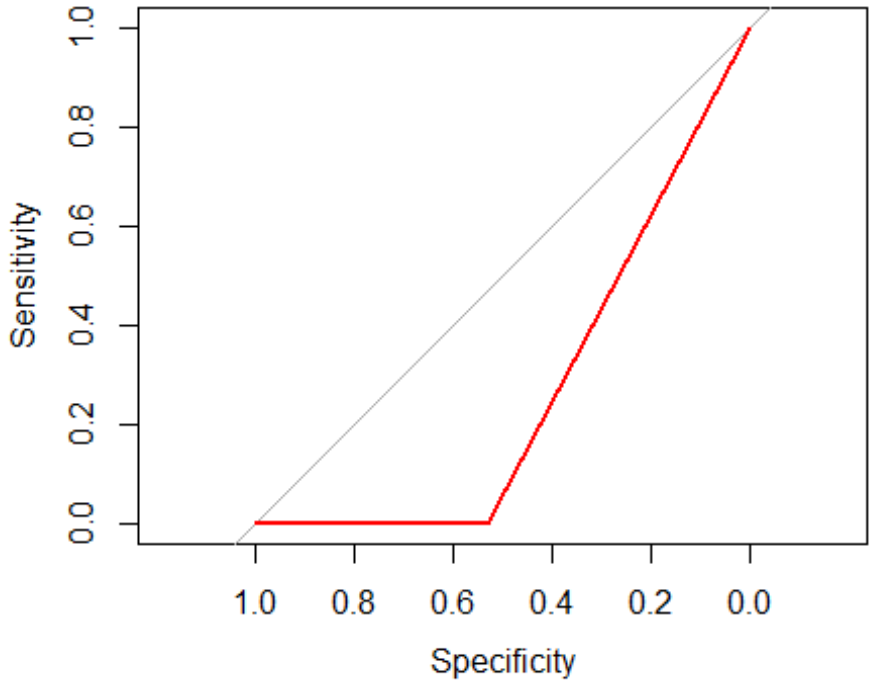
Método average:

```
#-------------------------------------------------------------------------
----------
#                           METHOD AVERAGE
#-------------------------------------------------------------------------
----------
#Dado un número de clústers k determinar la altura requerida
#para que tengamos el número de clúster k.
clustersD <- clusterJD(df, distancia, 1:2, "average", 2, name)
```



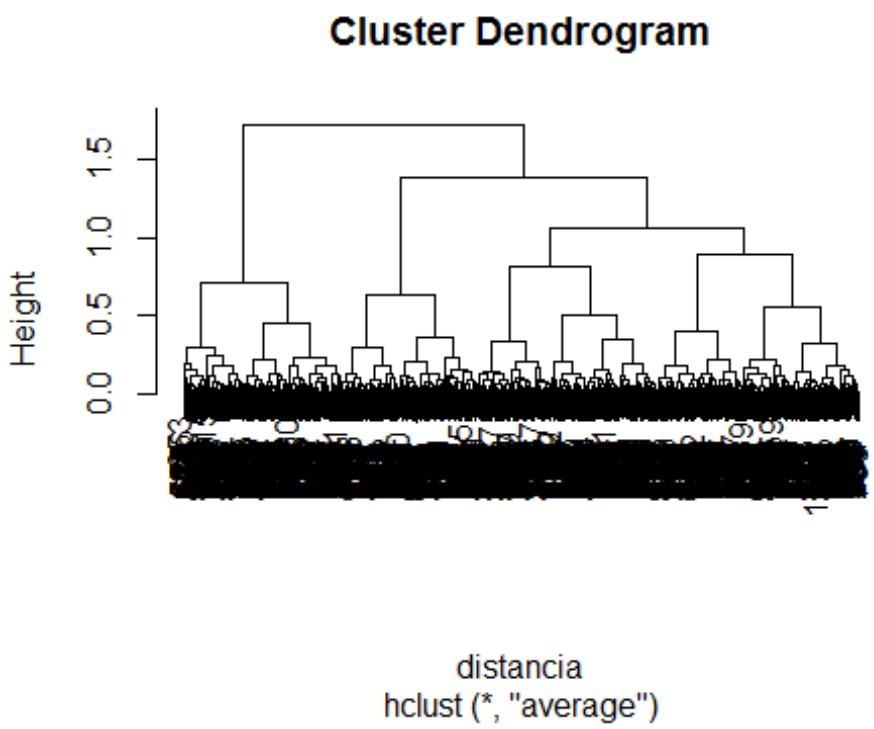

```
#Generamos la matriz de confusion
MatrixConfusionCJDA <- matrizconfusion(df$class, clustersD)
MatrixConfusionCJDA

##      Cluster
## Clase   1   2
##     1 234 266
##     2   0 500

#Calculamos la precision del modelo
PrecisionDA <- precision(MatrixConfusionCJDA)
PrecisionDA

## [1] 0.734
```

```r
#Generamos la curva de ROC
modeloDA <- roc(df$class, clustersD)
plot(modeloDA,type="l",col="red")
```

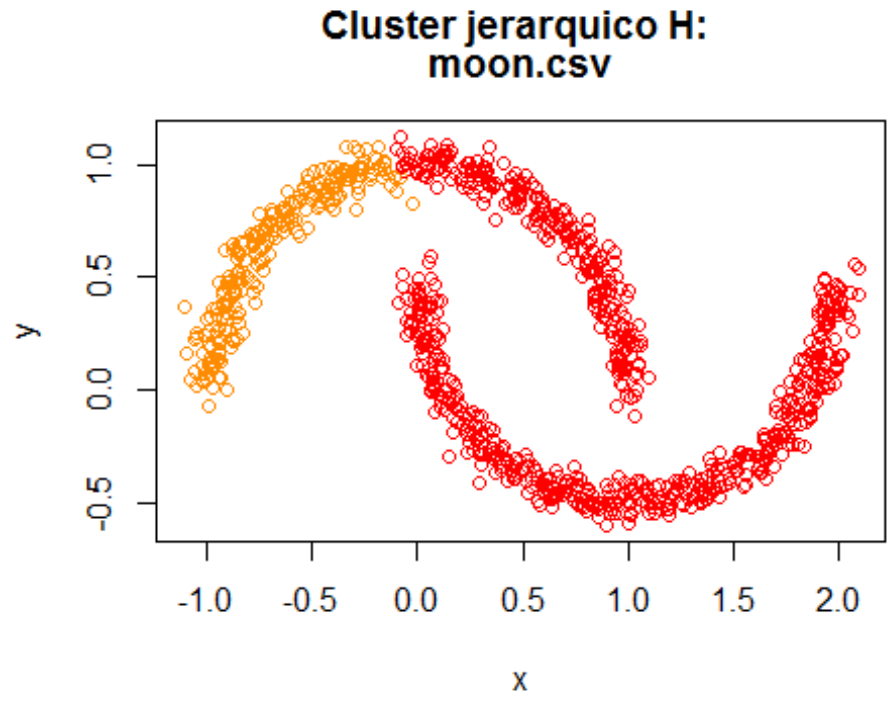



```
##
## Call:
## roc.default(response = df$class, predictor = clustersD)
##
## Data: clustersD in 500 controls (df$class 1) < 500 cases (df$class 2).
## Area under the curve: 0.266
```

```r
#*********************************************************
#Dada una altura h (una medida de disimilaridad) determinar
#el número de clústers que se obtienen.
clustersH <- clusterJH(df, distancia, 1:2, "average", 1.5, name)
```


Cluster Dendrogram
Height
0.0
0.5
1.0
1.5
distancia
hclust (*, "average")

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


Cluster jerarquico H:
moon.csv
y
1.0
0.5
0.0
-0.5
-1.0
-0.5
0.0
0.5
1.0
1.5
2.0
x

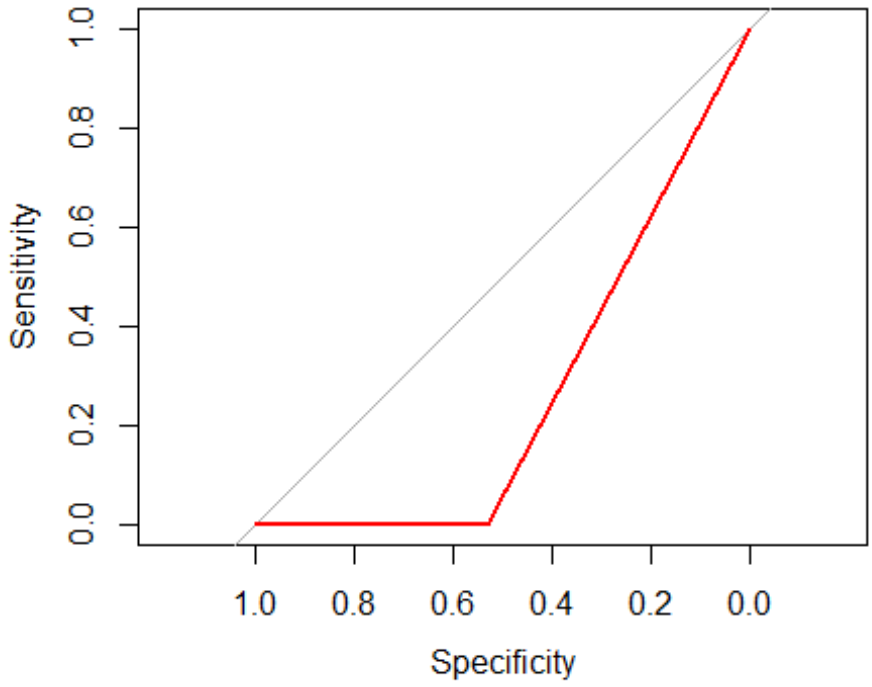
```r
#Generamos la matriz de confusion
MatrixConfusionCJHA <- matrizconfusion(df$class, clustersH)
MatrixConfusionCJHA
```

```
##      Cluster
## Clase   1   2
##     1 234 266
##     2   0 500
```

```r
#Calculamos la precision del modelo
PrecisionHA <- precision(MatrixConfusionCJHA)
PrecisionHA
```

```
## [1] 0.734
```

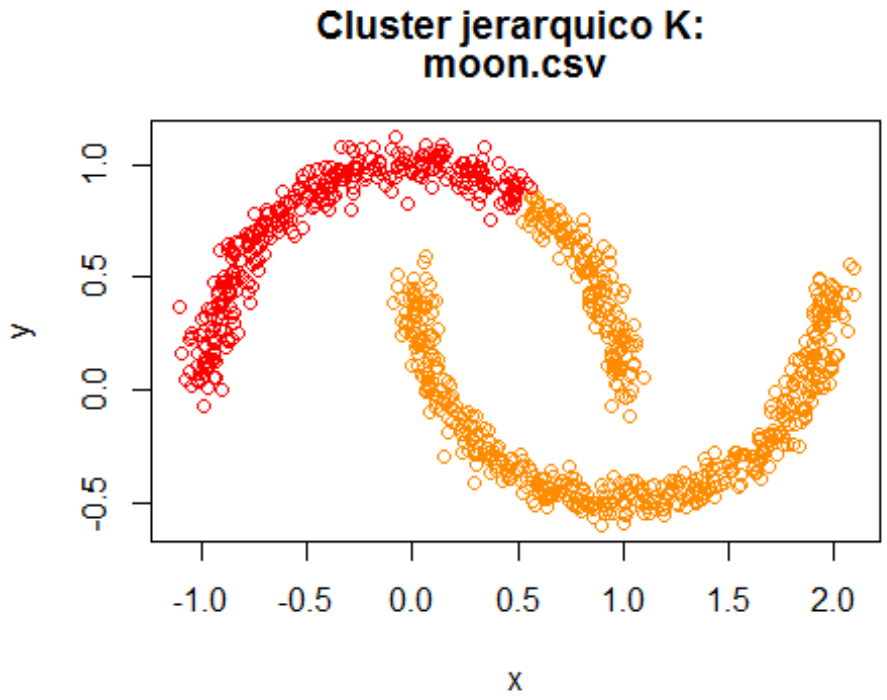
```r
#Generamos la curva de ROC
modeloHA <- roc(df$class, clustersH)
plot(modeloHA,type="l",col="red")
```




```
##
## Call:
## roc.default(response = df$class, predictor = clustersH)
##
## Data: clustersH in 500 controls (df$class 1) < 500 cases (df$class 2).
## Area under the curve: 0.266
```

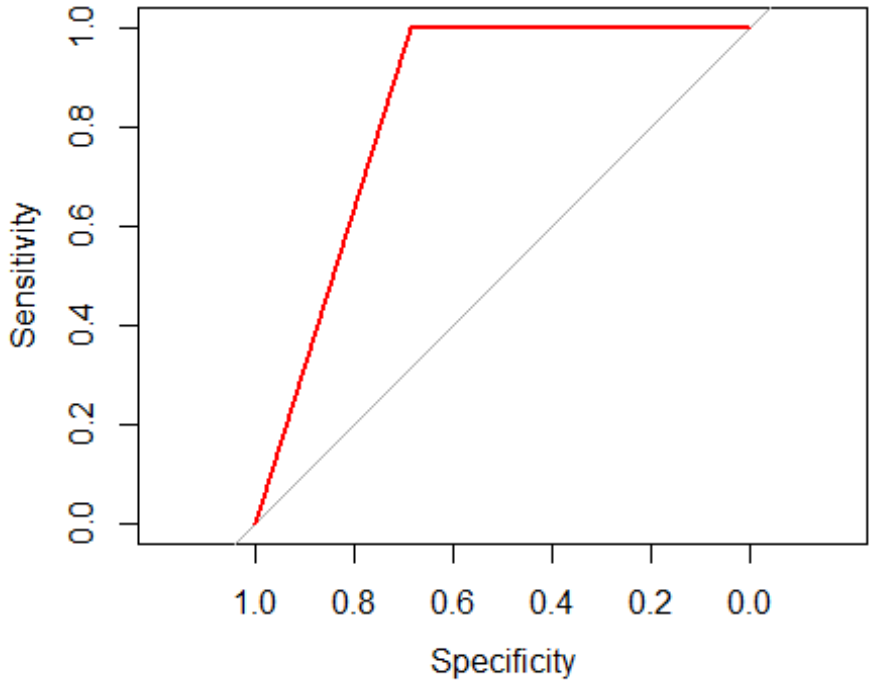
Método ward.D

```
#-------------------------------------------------------------------------
----------
#                          METHOD ward.D
#-------------------------------------------------------------------------
----------
#Dado un número de clústers k determinar la altura requerida
#para que tengamos el número de clúster k.
clustersD <- clusterJD(df, distancia, 1:2, "ward.D", 2, name)
```



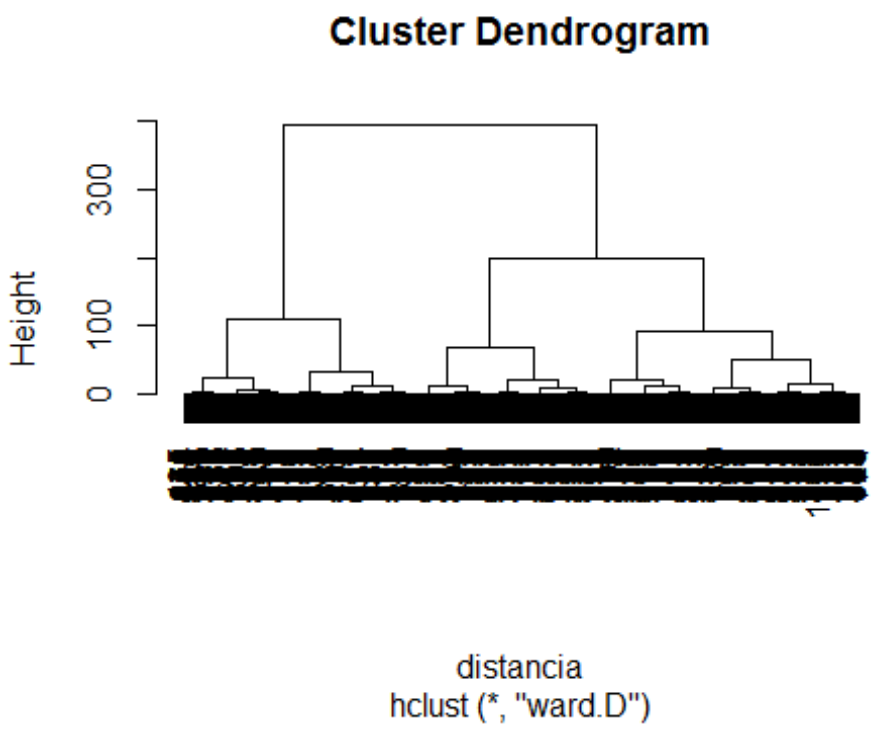

```
#Generamos la matriz de confusion
MatrixConfusionCJDW <- matrizconfusion(df$class, clustersD)
MatrixConfusionCJDW

##      Cluster
## Clase   1   2
##     1 344 156
##     2   0 500

#Calculamos la precision del modelo
PrecisionDW <- precision(MatrixConfusionCJDW)
PrecisionDW

## [1] 0.844
```

```r
#Generamos la curva de ROC
modeloDW <- roc(df$class, clustersD)
plot(modeloDW,type="l",col="red")
```

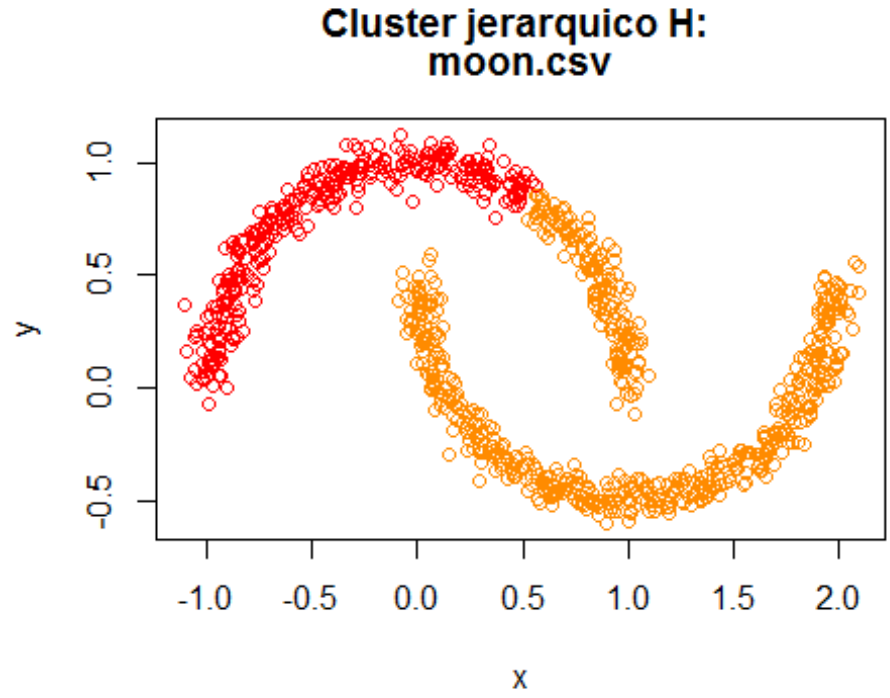



```
##
## Call:
## roc.default(response = df$class, predictor = clustersD)
##
## Data: clustersD in 500 controls (df$class 1) < 500 cases (df$class 2).
## Area under the curve: 0.844
```

```r
#*********************************************************
#Dada una altura h (una medida de disimilaridad) determinar
#el número de clústers que se obtienen.
clustersH <- clusterJH(df, distancia, 1:2, "ward.D", 300, name)
```


Cluster Dendrogram
Height
0
100
300
distancia
hclust (*, "ward.D")

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


Cluster jerarquico H:
moon.csv
y
-0.5
0.0
0.5
1.0
x
-1.0
-0.5
0.0
0.5
1.0
1.5
2.0

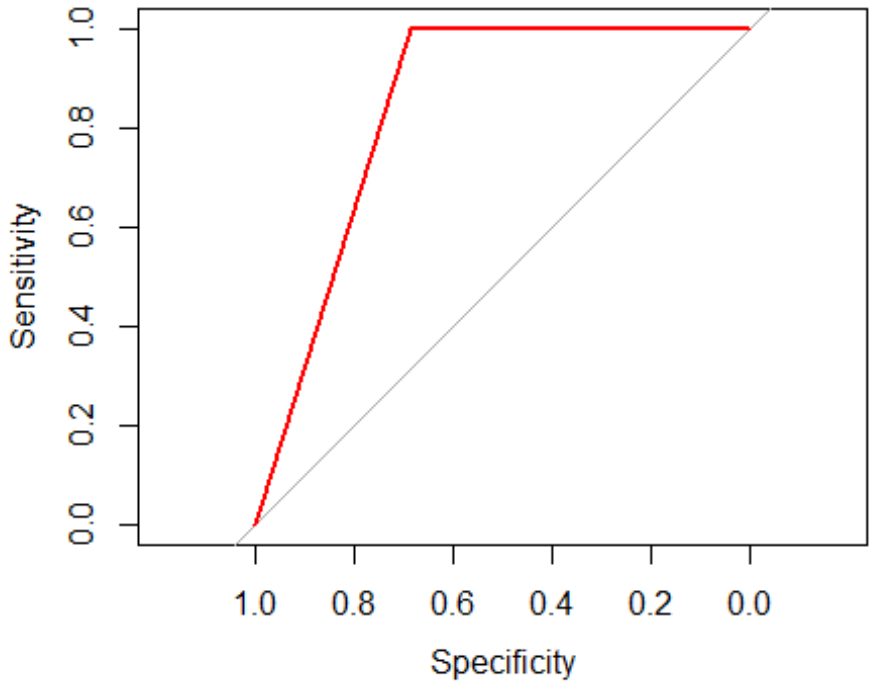
```r
#Generamos la matriz de confusion
MatrixConfusionCJHW <- matrizconfusion(df$class, clustersH)
MatrixConfusionCJHW
```

```
##      Cluster
## Clase   1   2
##     1 344 156
##     2   0 500
```

```r
#Calculamos la precision del modelo
PrecisionHW <- precision(MatrixConfusionCJHW)
PrecisionHW
```

```
## [1] 0.844
```

```r
#Generamos la curva de ROC
modeloHW <- roc(df$class, clustersH)
plot(modeloHW,type="l",col="red")
```




```
##
## Call:
## roc.default(response = df$class, predictor = clustersH)
##
## Data: clustersH in 500 controls (df$class 1) < 500 cases (df$class 2).
## Area under the curve: 0.844
```

Mejor modelo

```
#-------------------------------------------------------------------------
----------
#                                  MEJOR MODELO
#-------------------------------------------------------------------------
----------
precisiones <- c(PrecisionK, PrecisionDC, PrecisionHC, PrecisionDS,
PrecisionHS, PrecisionDA,
                 PrecisionHA, PrecisionDW, PrecisionHW)
x <- which.max(precisiones)
mejormodelo <- bestmodel(x)
cat("El mejor modelo es: ", mejormodelo, ", que posee una precision de: ",
precisiones[x], ".")

## El mejor modelo es:  CLASIFICACION JERARQUICA K: METHOD SINGLE , que posee
una precision de:  1 .
```

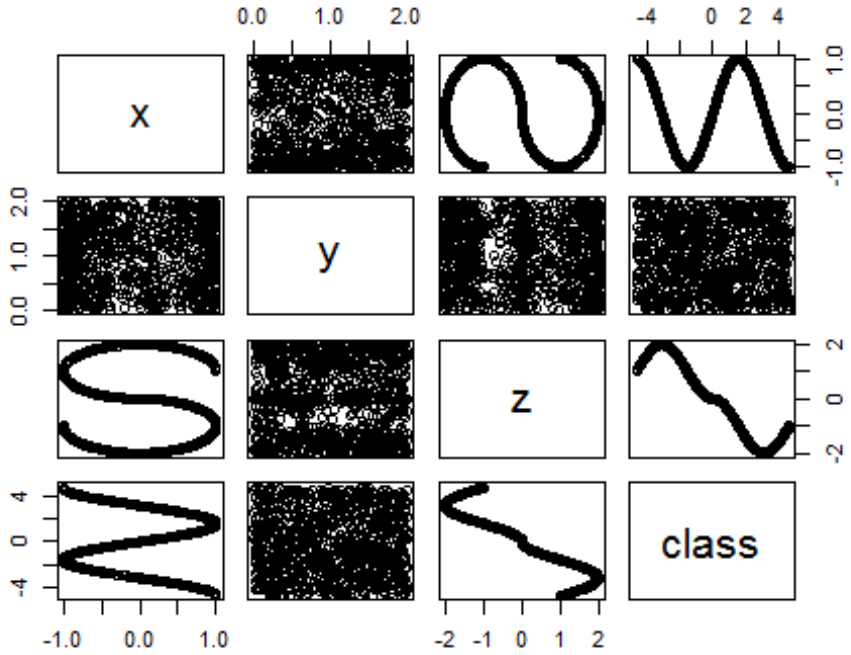
s.csv

- Preprocesamiento:

```
setwd("C:/Users/Eric/Desktop/AprendizajeNoSupervisado")
name = "help.csv"
#Lectura de datos.
df = read.csv(file =
"C:/Users/Eric/Desktop/AprendizajeNoSupervisado/data/s.csv", header = F)
#Modificamos el nombre de las columnas por comodidad.
colnames(df) <- c("x","y","z","class")
#Ordenamos la columna clase
df <- df[ order(df$class), ]
```

- Analisis exploratorio del dataset:

```
#*************************************************************************
************
#                    Analisis exploratorio del dataset
#*************************************************************************
************
#Matriz de dispersion
plot(df)
```



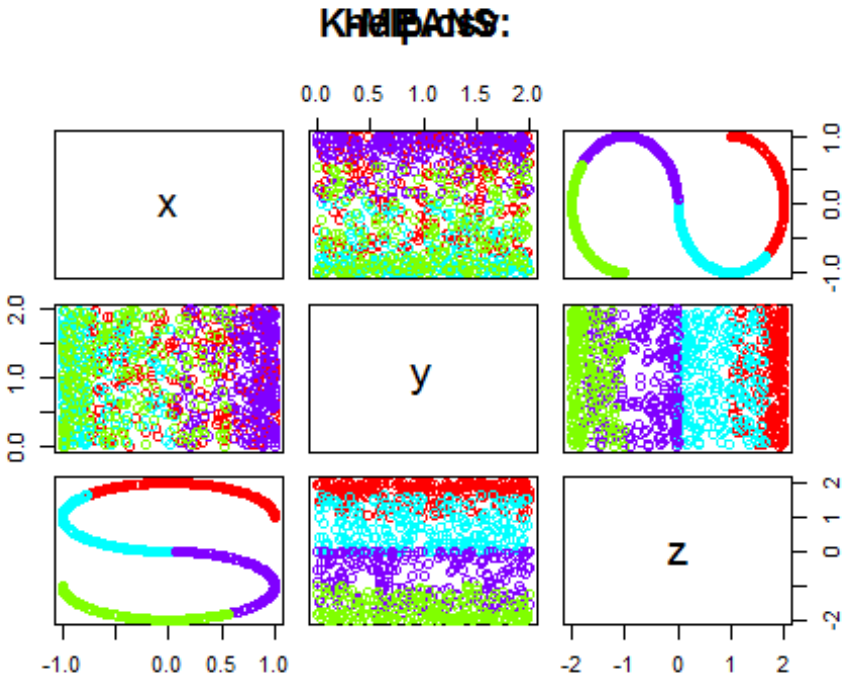
Reglas para asignar las clases: En este caso es complicado decidir cuantas clases hay, sin embargo por decisión personal elegí 4 clases.

```
#**********************REGLA PARA ASIGNAR CLASES*****************************
reglas <- function(x){
  if (x < -2.230634){
    return(1)
  }else if (x < -0.010020){
    return(2)
  }else if (x < 2.500632){
    return(3)
  }else{
    return(4)
  }
}

for (x in 1:nrow(df)) {

  df$class[x] <- reglas(df$class[x])
}
#Observamos cuantos elementos hay de cada clase.
table(df$class)

##
##   1   2   3   4
## 266 251 250 233
```

```
#1   2   3   4
#266 251 250 233

palette(rainbow(length(unique(df$class))))
plot3d(df$x, df$y, df$z, type = "s",size = 2, col = df$class)
```

K-medias:

```
#*************************************************************************************
************
#                                    K-MEDIAS
#*************************************************************************************
************
#Aplicamos k=4 ya que identificamos 4 conglomerados.
#kmedias(Dataframe, Columnas,K)
modeloK <- kmedias(df, 1:3, 4, name)
```



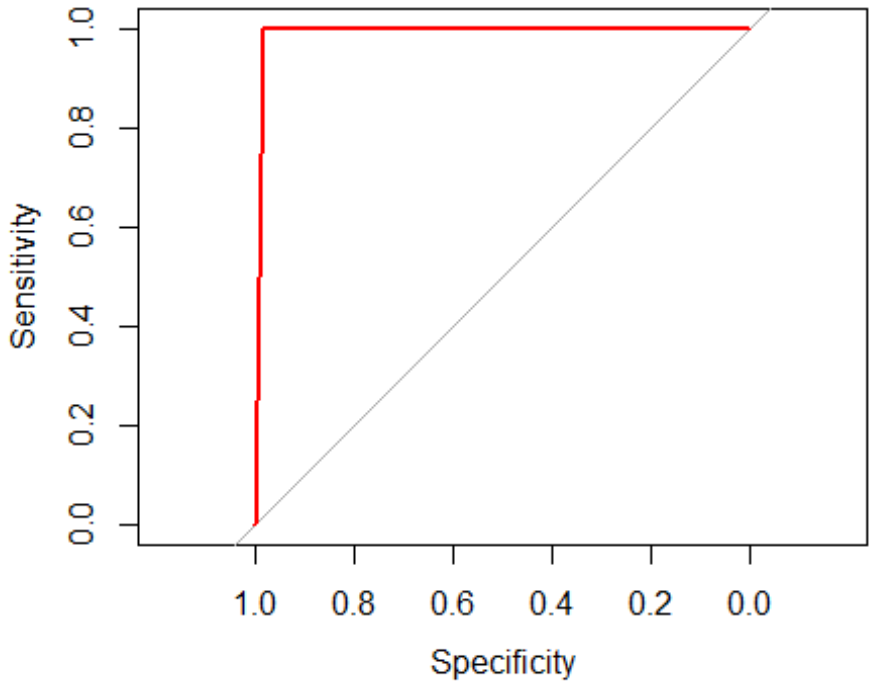

```
plot3d(df$x, df$y, df$z, type = "s",size = 2, col = modeloK$cluster)
#Generamos la matriz de confusion
MatrixConfusionK <- matrizconfusion(df$class,modeloK$cluster)
MatrixConfusionK

##      Cluster
## Clase   1   2   3   4
##     1 262   4   0   0
##     2   0 251   0   0
```

```
##     3   0   7 242   1
##     4   0   0   1 232
```

```
#Calculamos la precision del modelo
PrecisionK <- precision(MatrixConfusionK)
PrecisionK
```

```
## [1] 0.987
```

```
#Generamos la curva de ROC
modeloKROC <- roc(df$class, modeloK$cluster)
plot(modeloKROC,type="l",col="red")
```




```
##
## Call:
## roc.default(response = df$class, predictor = modeloK$cluster)
##
## Data: modeloK$cluster in 266 controls (df$class 1) < 251 cases (df$class
2).
## Area under the curve: 0.9925
```

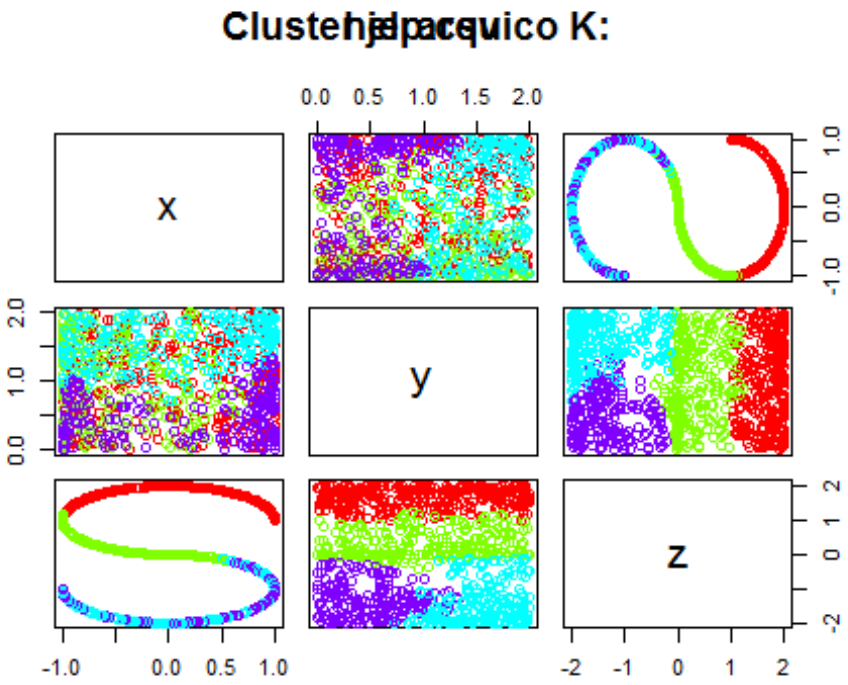
Clusters jerárquicos:

```
#*************************************************************************
***********
#                                    Cluster Jerarquicos
#*************************************************************************
```

```
************
#Copy del dataset
datos = df
#Elimino la columna clase para realizar aprendizaje no supervisado.
datos$class <- NULL
#Convierto el dataframe en una matriz
datos= as.matrix(datos)
#Calculamos la matriz de distancia
distancia = dist(datos)
```

Método complete:

```
#-------------------------------------------------------------------------------
----------
#                                   METHOD COMPLETE
#-------------------------------------------------------------------------------
----------
#Dado un número de clústers k determinar la altura requerida
#para que tengamos el número de clúster k.
clustersD <- clusterJD(df, distancia, 1:3, "complete", 4, name)
```



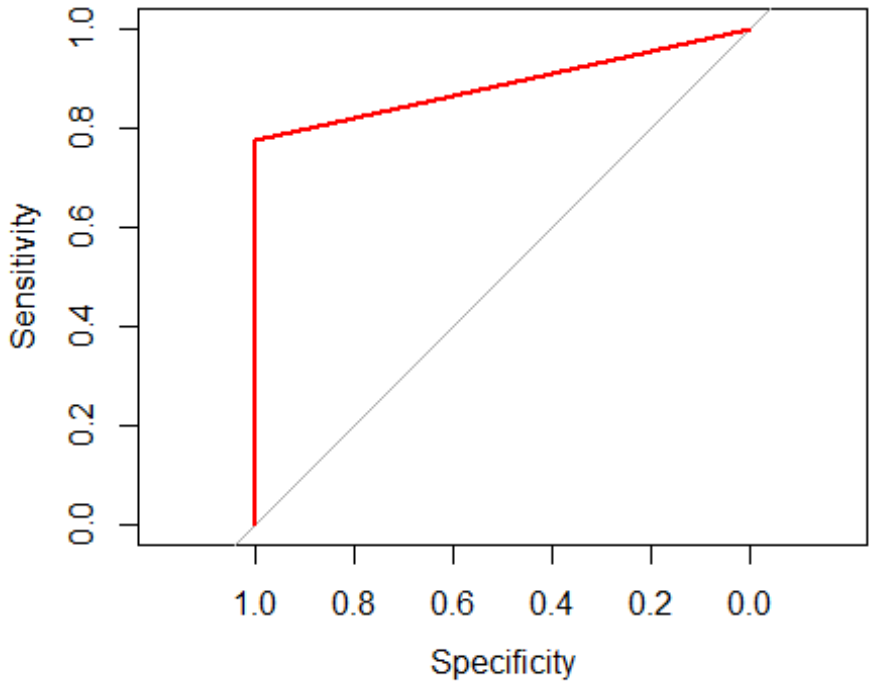

```
#Generamos la matriz de confusion
plot3d(df$x, df$y, df$z, type = "s",size = 2, col = clustersD)
MatrixConfusionCJDC <- matrizconfusion(df$class, clustersD)
MatrixConfusionCJDC
```

```
##      Cluster
## Clase   1   2   3   4
##     1 266   0   0   0
##     2  57 194   0   0
##     3   0  60 108  82
##     4   0   0  97 136
```

```
#Calculamos la precision del modelo
PrecisionDC <- precision(MatrixConfusionCJDC)
PrecisionDC
```

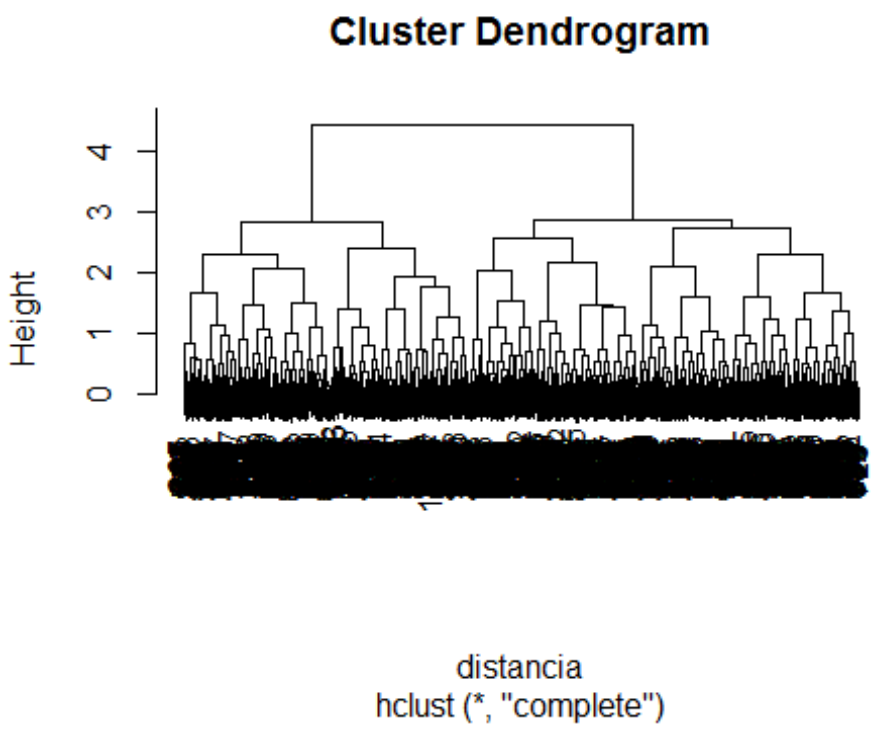
```
## [1] 0.704
```

```
#Generamos la curva de ROC
modeloDC <- roc(df$class, clustersD)
plot(modeloDC,type="l",col="red")
```

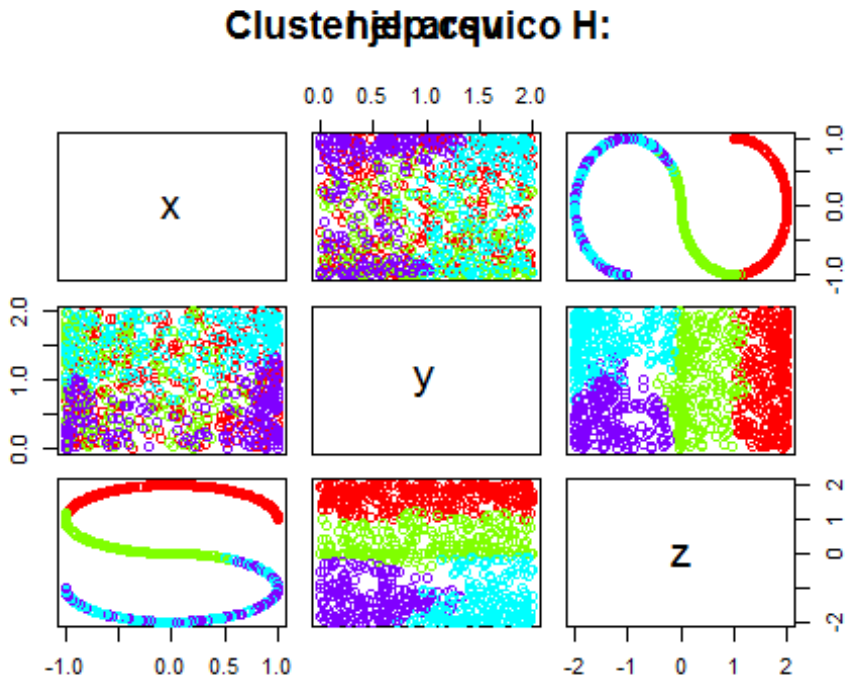



```
##
## Call:
## roc.default(response = df$class, predictor = clustersD)
##
## Data: clustersD in 266 controls (df$class 1) < 251 cases (df$class 2).
## Area under the curve: 0.8865
```

```
#*********************************************************
#Dada una altura h (una medida de disimilaridad) determinar
#el número de clústers que se obtienen.
clustersH <- clusterJH(df, distancia, 1:3, "complete", 2.8, name)
```




```
## [1] 1 2 3 4
```




```
plot3d(df$x, df$y, df$z, type = "s",size = 2, col = clustersH)
#Generamos la matriz de confusion
```

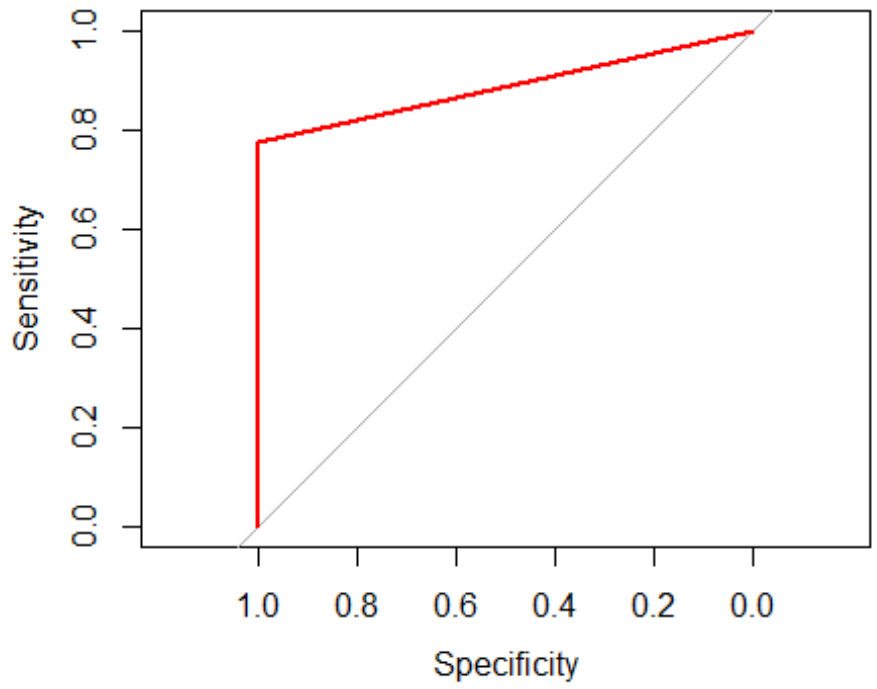
```r
MatrixConfusionCJHC <- matrizconfusion(df$class, clustersH)
MatrixConfusionCJHC
```

```
##      Cluster
## Clase   1   2   3   4
##     1 266   0   0   0
##     2  57 194   0   0
##     3   0  60 108  82
##     4   0   0  97 136
```

```r
#Calculamos la precision del modelo
PrecisionHC <- precision(MatrixConfusionCJHC)
PrecisionHC
```

```
## [1] 0.704
```

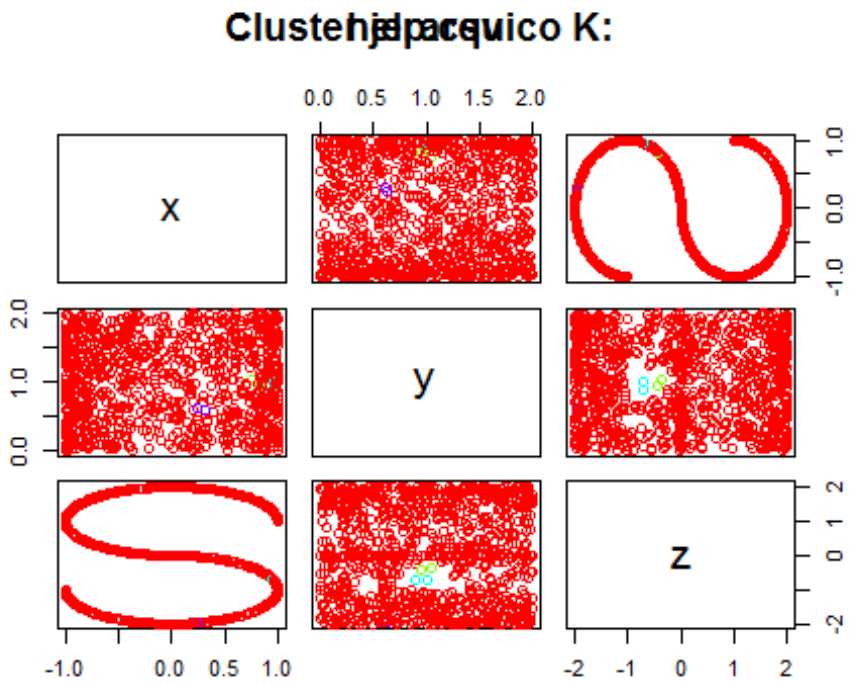
```r
#Generamos la curva de ROC
modeloHC <- roc(df$class, clustersH)
plot(modeloHC,type="l",col="red")
```




```
##
## Call:
## roc.default(response = df$class, predictor = clustersH)
##
## Data: clustersH in 266 controls (df$class 1) < 251 cases (df$class 2).
## Area under the curve: 0.8865
```

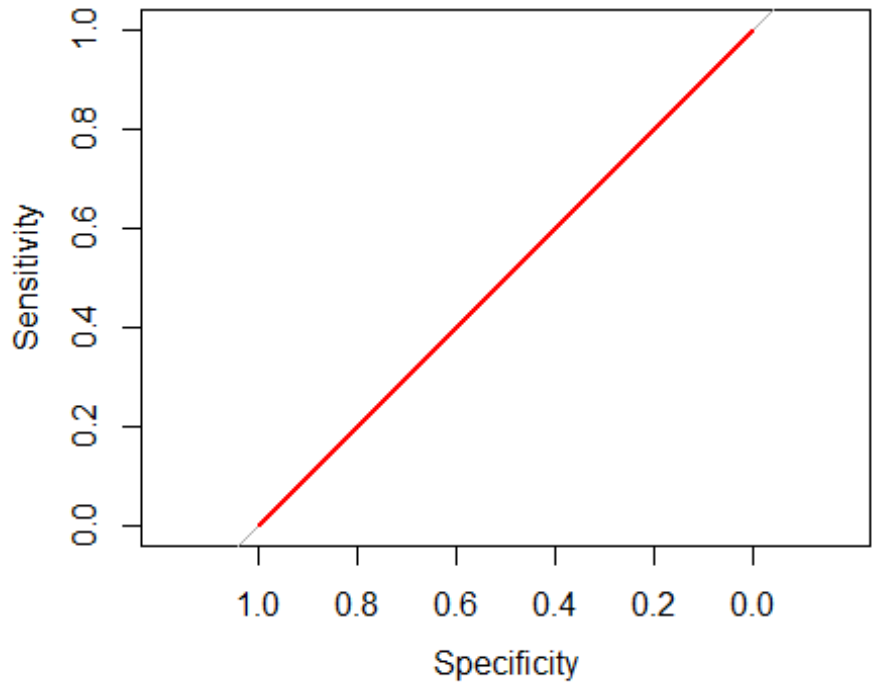
Método single:

```
#-------------------------------------------------------------------------
----------
#                              METHOD SINGLE
#-------------------------------------------------------------------------
----------
#Dado un número de clústers k determinar la altura requerida
#para que tengamos el número de clúster k.
clustersD <- clusterJD(df, distancia, 1:3, "single", 4, name)
```



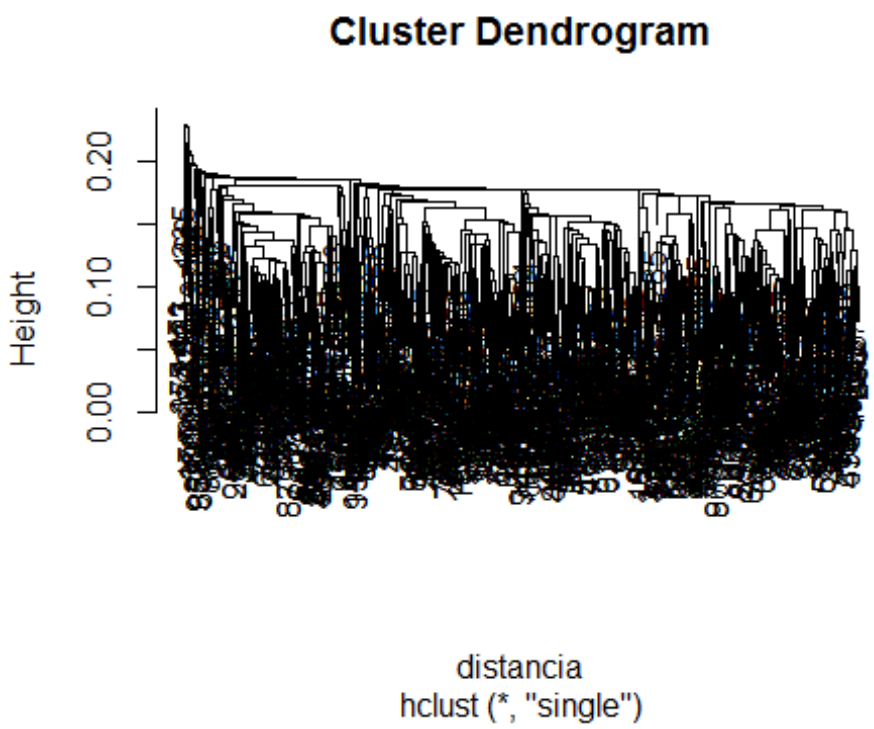

```
plot3d(df$x, df$y, df$z, type = "s",size = 2, col = clustersD)
#Generamos la matriz de confusion
MatrixConfusionCJDS <- matrizconfusion(df$class, clustersD)
MatrixConfusionCJDS

##      Cluster
## Clase   1   2   3   4
##     1 266   0   0   0
##     2 251   0   0   0
##     3 246   2   2   0
##     4 231   0   0   2

#Calculamos la precision del modelo
PrecisionDS <- precision(MatrixConfusionCJDS)
PrecisionDS

## [1] 0.27
```

```r
#Generamos la curva de ROC
modeloDS <- roc(df$class, clustersD)
plot(modeloDS,type="l",col="red")
```

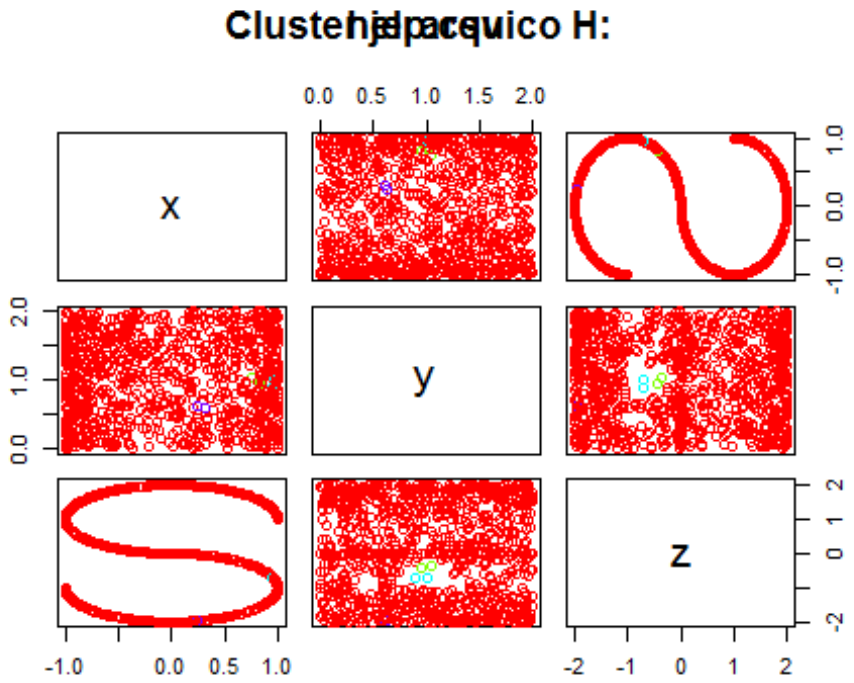



```
##
## Call:
## roc.default(response = df$class, predictor = clustersD)
##
## Data: clustersD in 266 controls (df$class 1) < 251 cases (df$class 2).
## Area under the curve: 0.5
```

```r
#*********************************************************
#Dada una altura h (una medida de disimilaridad) determinar
#el número de clústers que se obtienen.
clustersH <- clusterJH(df, distancia, 1:3, "single", 0.21, name)
```




```
## [1] 1 2 3 4
```




```
plot3d(df$x, df$y, df$z, type = "s",size = 2, col = clustersH)
#Generamos la matriz de confusion
```

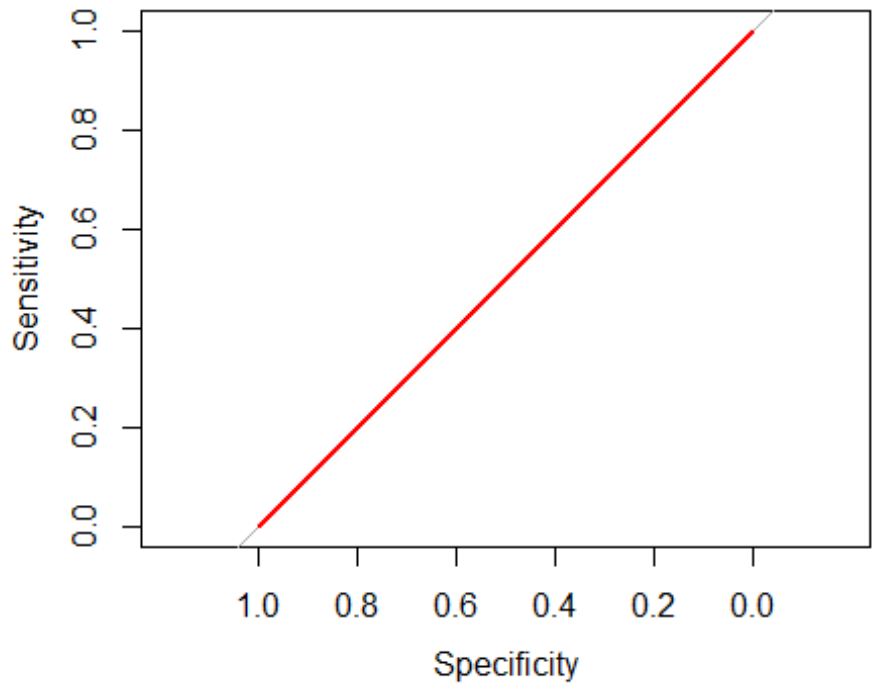
```
MatrixConfusionCJHS <- matrizconfusion(df$class, clustersH)
MatrixConfusionCJHS
```

```
##      Cluster
## Clase   1   2   3   4
##     1 266   0   0   0
##     2 251   0   0   0
##     3 246   2   2   0
##     4 231   0   0   2
```

```
#Calculamos la precision del modelo
PrecisionHS <- precision(MatrixConfusionCJHS)
PrecisionHS
```

```
## [1] 0.27
```

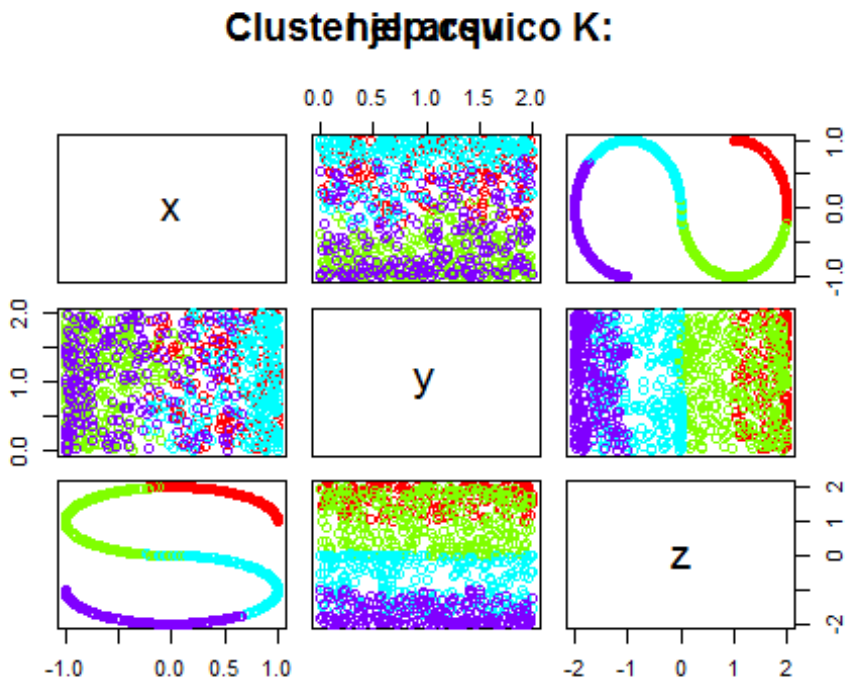
```
#Generamos la curva de ROC
modeloHS <- roc(df$class, clustersH)
plot(modeloHS,type="l",col="red")
```




```
##
## Call:
## roc.default(response = df$class, predictor = clustersH)
##
## Data: clustersH in 266 controls (df$class 1) < 251 cases (df$class 2).
## Area under the curve: 0.5
```

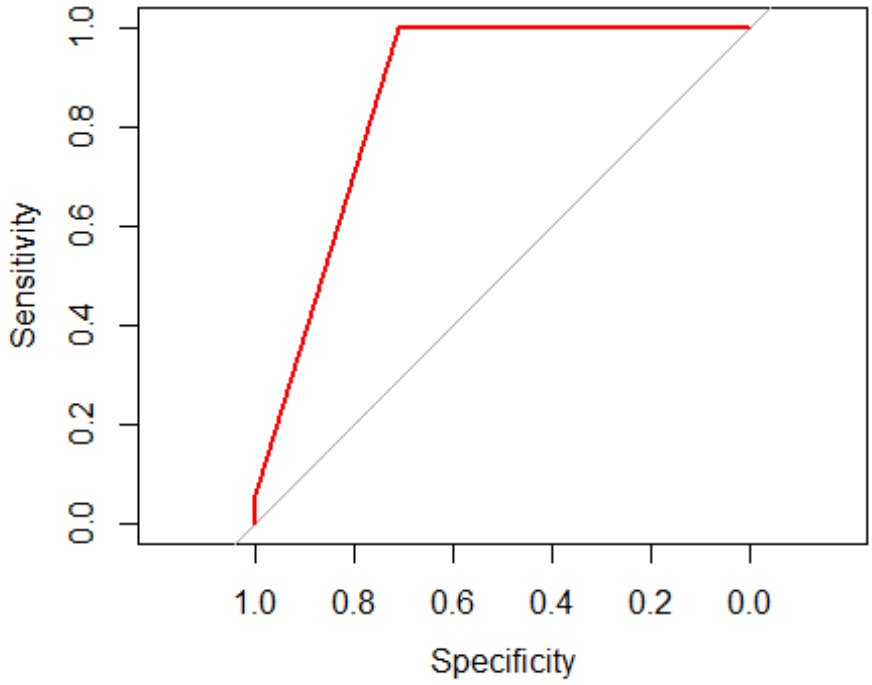
Método average:

```
#-------------------------------------------------------------------------
----------
#                              METHOD AVERAGE
#-------------------------------------------------------------------------
----------
#Dado un número de clústers k determinar la altura requerida
#para que tengamos el número de clúster k.
clustersD <- clusterJD(df, distancia, 1:3, "average", 4, name)
```



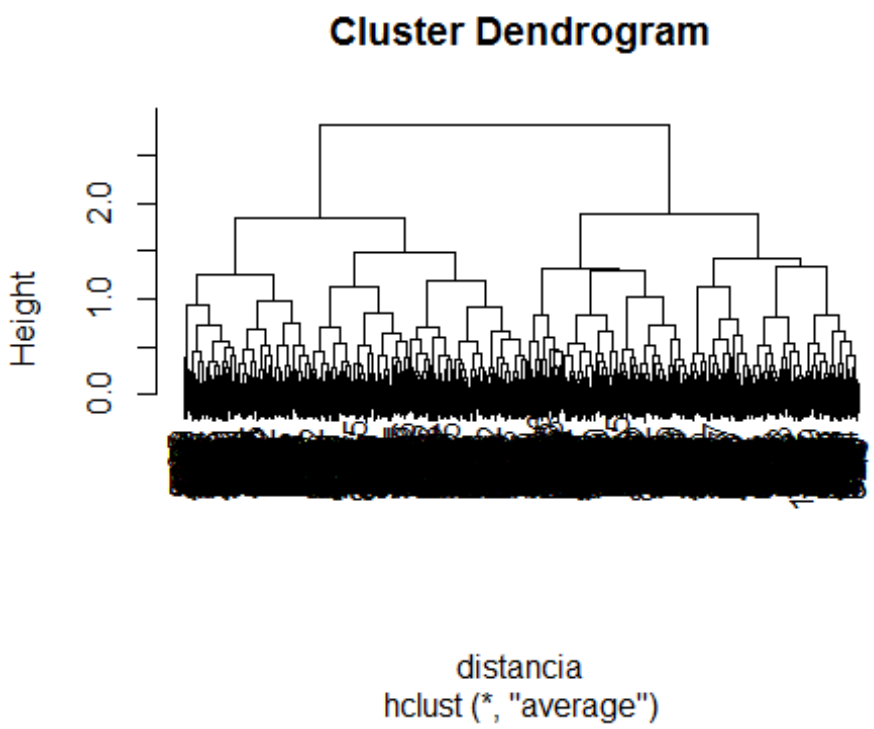

```
plot3d(df$x, df$y, df$z, type = "s",size = 2, col = clustersD)
#Generamos la matriz de confusion
MatrixConfusionCJDA <- matrizconfusion(df$class, clustersD)
MatrixConfusionCJDA

##      Cluster
## Clase   1   2   3   4
##     1 189  77   0   0
##     2   0 237  14   0
##     3   0   7 238   5
##     4   0   0   0 233

#Calculamos la precision del modelo
PrecisionDA <- precision(MatrixConfusionCJDA)
PrecisionDA

## [1] 0.897
```

```r
#Generamos la curva de ROC
modeloDA <- roc(df$class, clustersD)
plot(modeloDA,type="l",col="red")
```

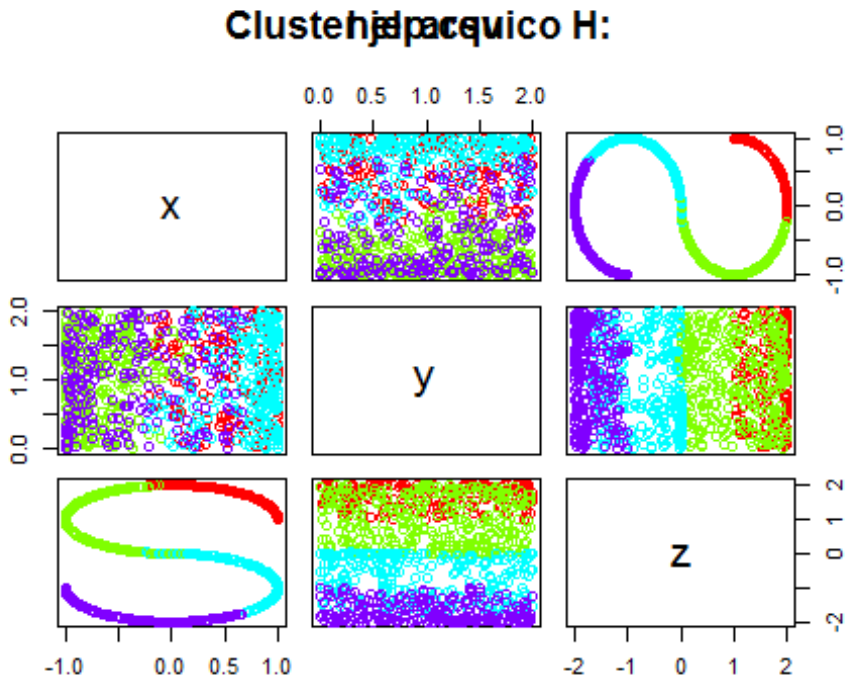



```
##
## Call:
## roc.default(response = df$class, predictor = clustersD)
##
## Data: clustersD in 266 controls (df$class 1) < 251 cases (df$class 2).
## Area under the curve: 0.8633
```

```r
#*********************************************************
#Dada una altura h (una medida de disimilaridad) determinar
#el número de clústers que se obtienen.
clustersH <- clusterJH(df, distancia, 1:3, "average", 1.5, name)
```




```
## [1] 1 2 3 4
```




```
plot3d(df$x, df$y, df$z, type = "s",size = 2, col = clustersH)
#Generamos la matriz de confusion
```

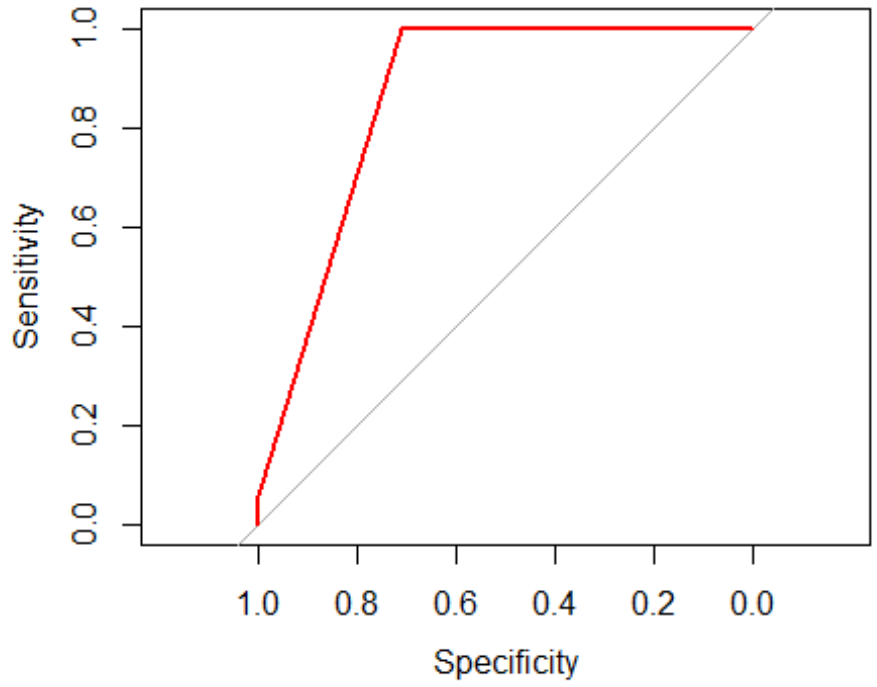
```r
MatrixConfusionCJHA <- matrizconfusion(df$class, clustersH)
MatrixConfusionCJHA
```

```
##      Cluster
## Clase   1   2   3   4
##     1 189  77   0   0
##     2   0 237  14   0
##     3   0   7 238   5
##     4   0   0   0 233
```

```r
#Calculamos la precision del modelo
PrecisionHA <- precision(MatrixConfusionCJHA)
PrecisionHA
```

```
## [1] 0.897
```

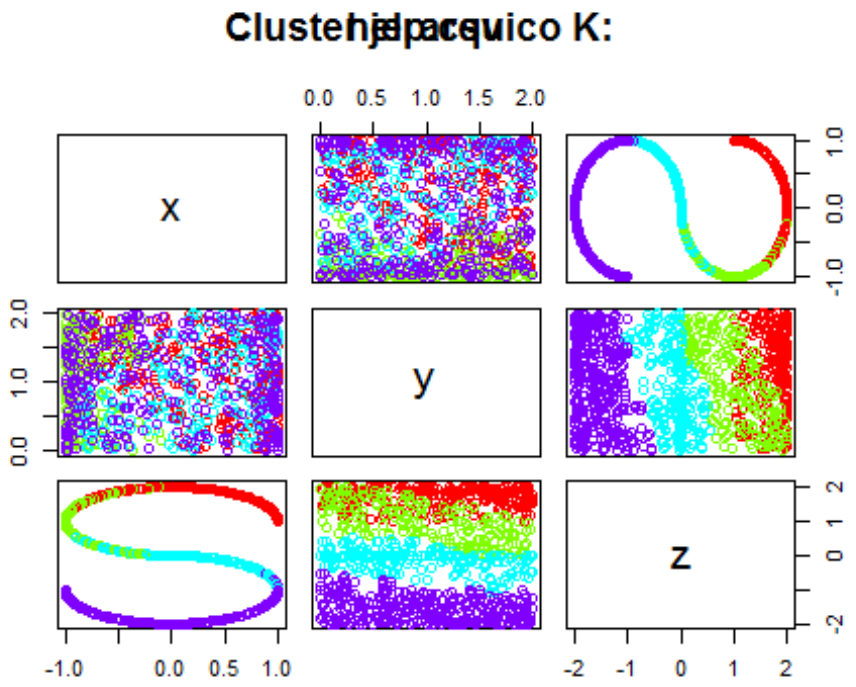
```r
#Generamos la curva de ROC
modeloHA <- roc(df$class, clustersH)
plot(modeloHA,type="l",col="red")
```




```
##
## Call:
## roc.default(response = df$class, predictor = clustersH)
##
## Data: clustersH in 266 controls (df$class 1) < 251 cases (df$class 2).
## Area under the curve: 0.8633
```

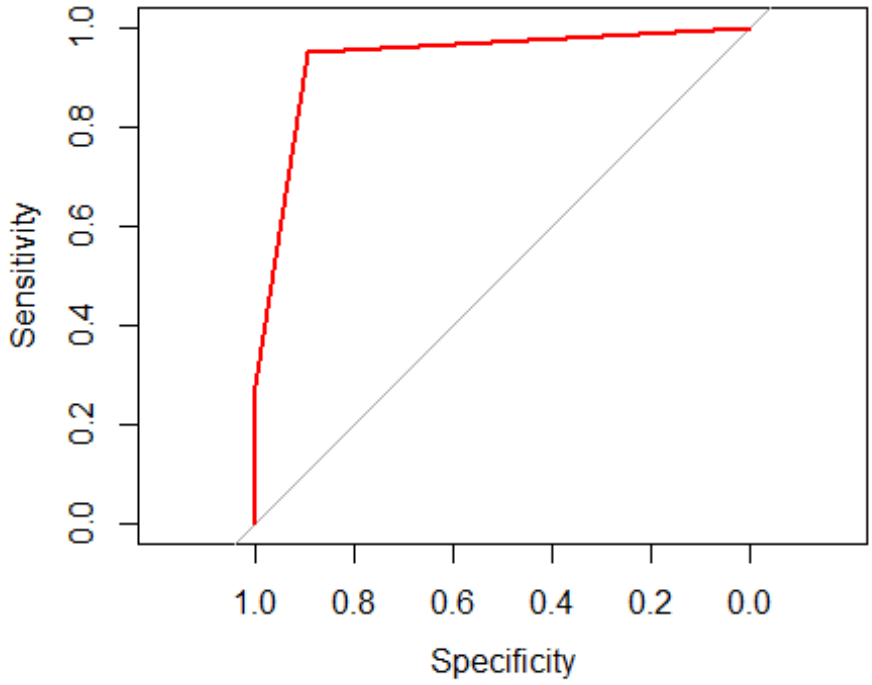
Método ward.D

```
#-------------------------------------------------------------------------
----------
#                                METHOD ward.D
#-------------------------------------------------------------------------
----------
#Dado un número de clústers k determinar la altura requerida
#para que tengamos el número de clúster k.
clustersD <- clusterJD(df, distancia, 1:3, "ward.D", 4, name)
```



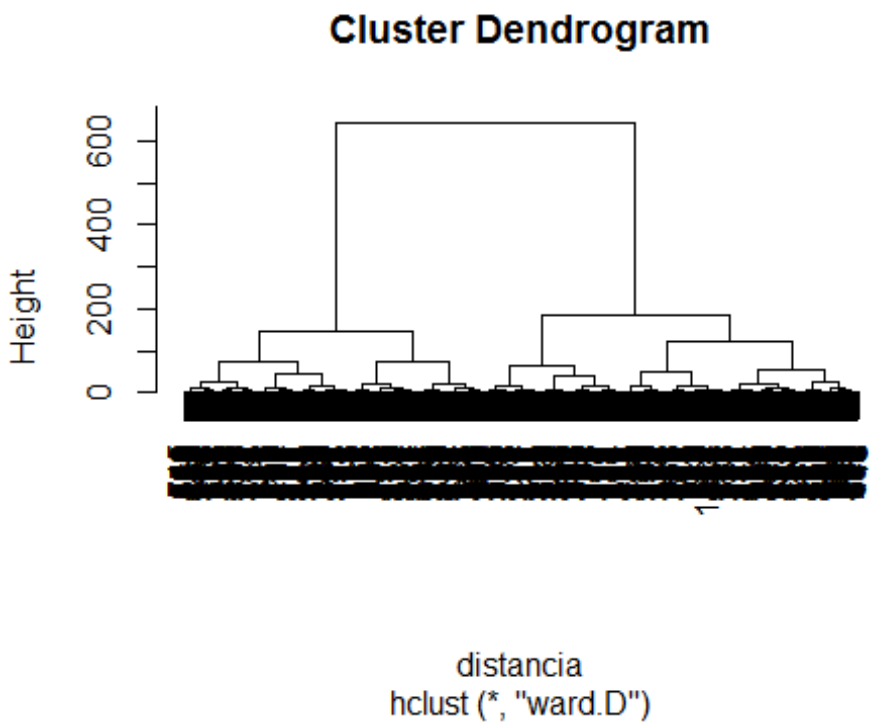

```
plot3d(df$x, df$y, df$z, type = "s",size = 2, col = clustersD)
#Generamos la matriz de confusion
MatrixConfusionCJDW <- matrizconfusion(df$class, clustersD)
MatrixConfusionCJDW

##      Cluster
## Clase   1   2   3   4
##     1 238  28   0   0
##     2  13 168  70   0
##     3   0   0 134 116
##     4   0   0   0 233

#Calculamos la precision del modelo
PrecisionDW <- precision(MatrixConfusionCJDW)
PrecisionDW

## [1] 0.773
```

```r
#Generamos la curva de ROC
modeloDW <- roc(df$class, clustersD)
plot(modeloDW,type="l",col="red")
```

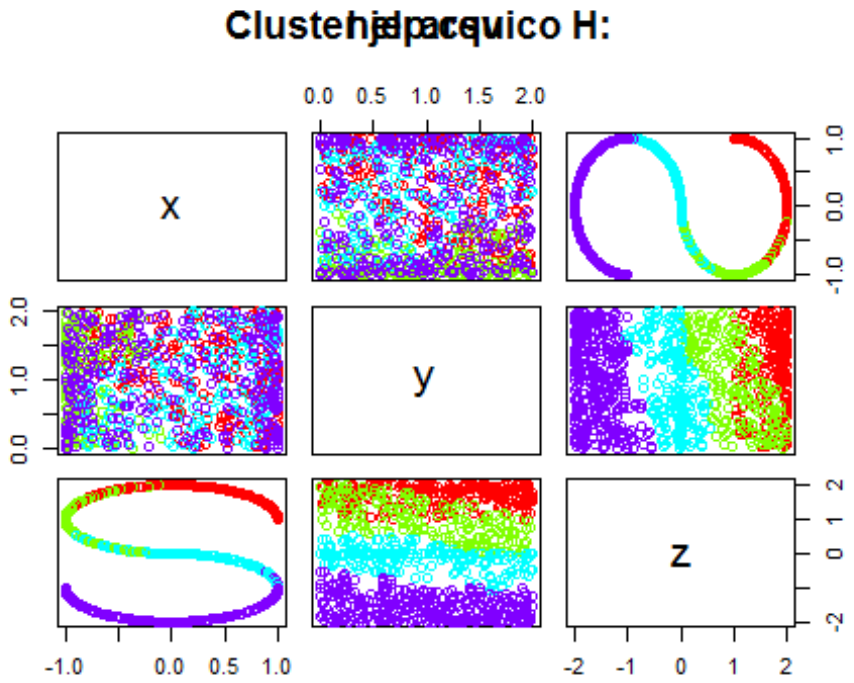



```
##
## Call:
## roc.default(response = df$class, predictor = clustersD)
##
## Data: clustersD in 266 controls (df$class 1) < 251 cases (df$class 2).
## Area under the curve: 0.9362
```

```r
#*********************************************************
#Dada una altura h (una medida de disimilaridad) determinar
#el número de clústers que se obtienen.
clustersH <- clusterJH(df, distancia, 1:3, "ward.D", 140, name)
```




```
## [1] 1 2 3 4
```




```
plot3d(df$x, df$y, df$z, type = "s",size = 2, col = clustersH)
#Generamos la matriz de confusion
```

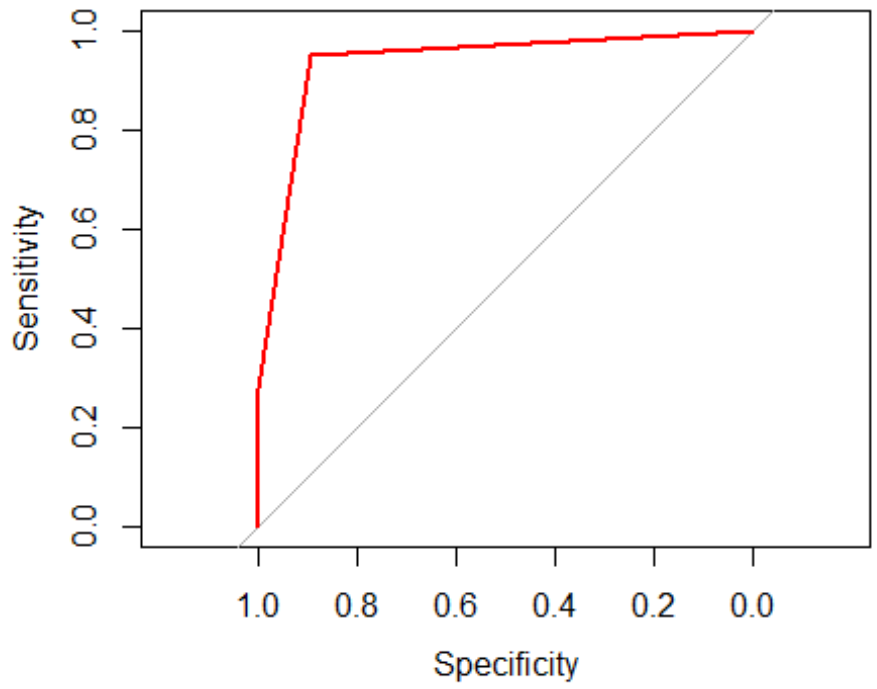
```r
MatrixConfusionCJHW <- matrizconfusion(df$class, clustersH)
MatrixConfusionCJHW
```

```
##      Cluster
## Clase   1   2   3   4
##     1 238  28   0   0
##     2  13 168  70   0
##     3   0   0 134 116
##     4   0   0   0 233
```

```r
#Calculamos la precision del modelo
PrecisionHW <- precision(MatrixConfusionCJHW)
PrecisionHW
```

```
## [1] 0.773
```

```r
#Generamos la curva de ROC
modeloHW <- roc(df$class, clustersH)
plot(modeloHW,type="l",col="red")
```




```
##
## Call:
## roc.default(response = df$class, predictor = clustersH)
##
## Data: clustersH in 266 controls (df$class 1) < 251 cases (df$class 2).
## Area under the curve: 0.9362
```

Mejor modelo

```
#-------------------------------------------------------------------------
----------
#                                   MEJOR MODELO
#-------------------------------------------------------------------------
----------
precisiones <- c(PrecisionK, PrecisionDC, PrecisionHC, PrecisionDS,
PrecisionHS, PrecisionDA,
                 PrecisionHA, PrecisionDW, PrecisionHW)
x <- which.max(precisiones)
mejormodelo <- bestmodel(x)
cat("El mejor modelo es: ", mejormodelo, ", que posee una precision de: ",
precisiones[x], ".")

## El mejor modelo es:  K-MEDIAS , que posee una precision de:  0.987 .
```