## Introducción

En el presente trabajo del máster en ciencia de datos e ingeniería de computadores, en el curso de Minería de datos: Aprendizaje no supervisado y detección de anomalías se desarrolla una aplicación de diferentes técnicas de agrupamiento sobre el dataset newthyroid, obtenido en [Keel.ugr.es](https://sci2s.ugr.es/keel/category.php?cat=uns) y que contiene la información de 215 personas clasificadas entre personas normales, personas sufriendo de hipotiroidismo y personas con hipertiroidismo. Se decide utilizar este dataset por ser un problema académico donde es importante aplicar métricas de calidad y comparación, por lo cual es necesario tener acceso a una clase para cada uno de los puntos. Serán aplicadas las técnicas de agrupamiento estudiadas durante el máster, las cuales son:

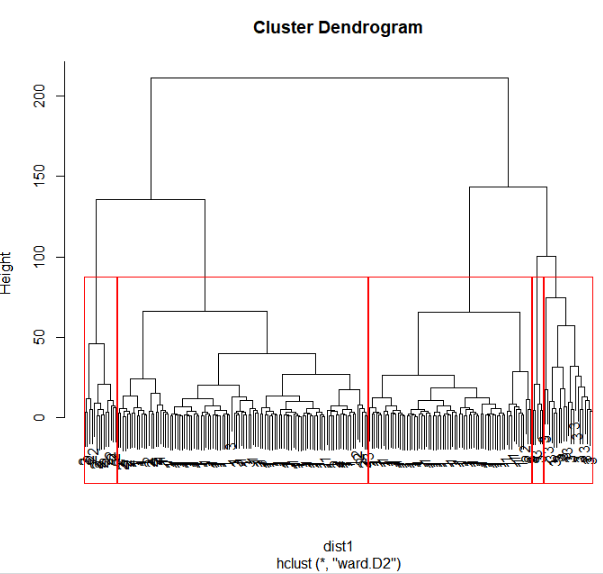
* Distancias jerárquico
* Distancias K-medias
* Distancias K-medoides
* DBScan
* K-medias difuso normalizado
* Jerárquico normalizado
* K-medias normalizado
* Medoides normalizado
* Medoides normalizado con valor óptimo de grupos

El documento contiene la aplicación de cada una de estas técnicas sobre el dataset mencionado con diferentes valores y un análisis de los resultados obtenidos en cada caso, en dependencia de los scripts que los contienen y la lógica relacionada a los mismos.

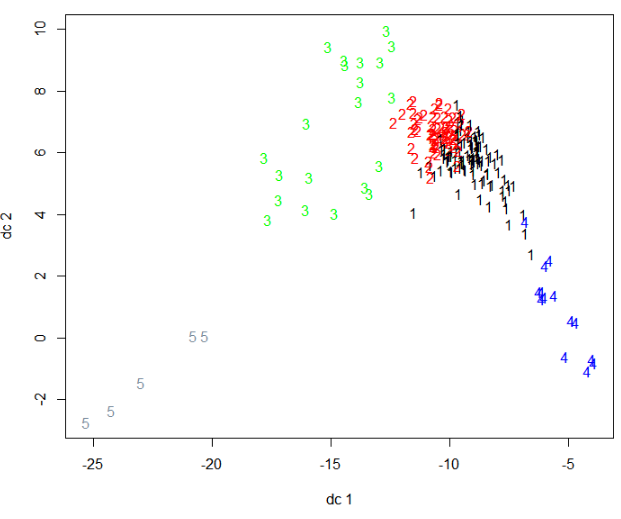
## Distancias jerárquico

Los métodos de agrupamiento jerárquicos se basan en construir un clúster anidado donde cada pieza del mismo es un clúster a su vez, estas piezas se van añadiendo al clúster principal basándose en la distancia entre ellos, típicamente usando la distancia euclídea, aunque puede ser provechoso usar otras mediciones de distancia según el problema. Además de la fórmula utilizada para medir la distancia entre puntos, es importante determinar la forma de medir la distancia entre los diferentes clúster, ya que de esto depende significativamente el resultado final de la agrupación. Métodos comunes son buscar la distancia entre los dos puntos más cercanos, entre los dos más lejanos y entre los centroides, aunque existen otras formas de medir la distancia.

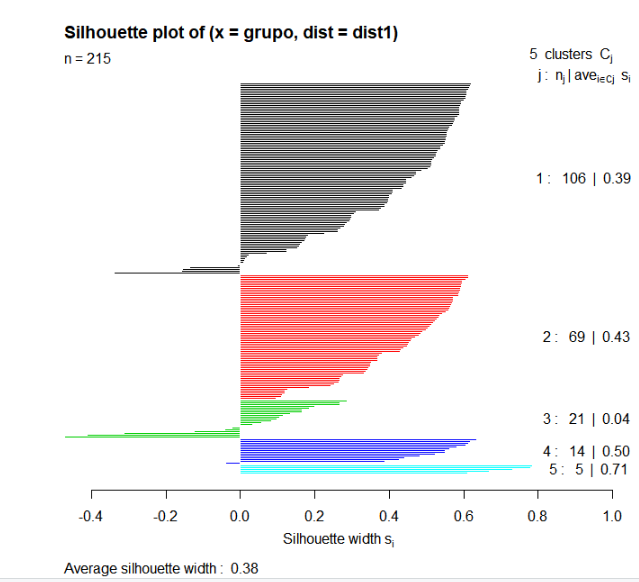
En el dataset en cuestión se va a utilizar la distancia euclídea, lo cual nos genera el siguiente dendograma que representa la jerarquía del clúster.



Los rectángulos representan cuales serían los puntos contenidos en caso de querer obtener 5 clústeres. Su representación en un plano es la siguiente:

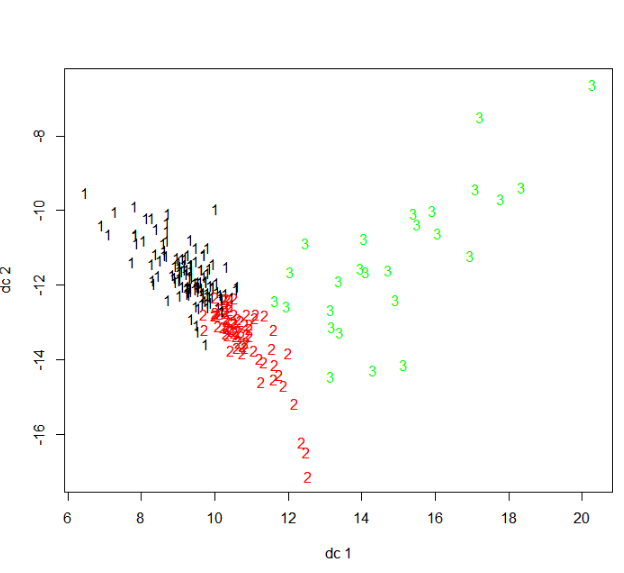


Realmente en este caso vemos que parte de los clústeres 1,2,3 y 4 se entremezclan bastante, dando señal de una baja cohesión en los clústeres formados. Esto es inevitable puesto que se definió que fueran creados 5 clústeres en un dataset que contiene solo 3 clases objetivo, y la silueta resultante:

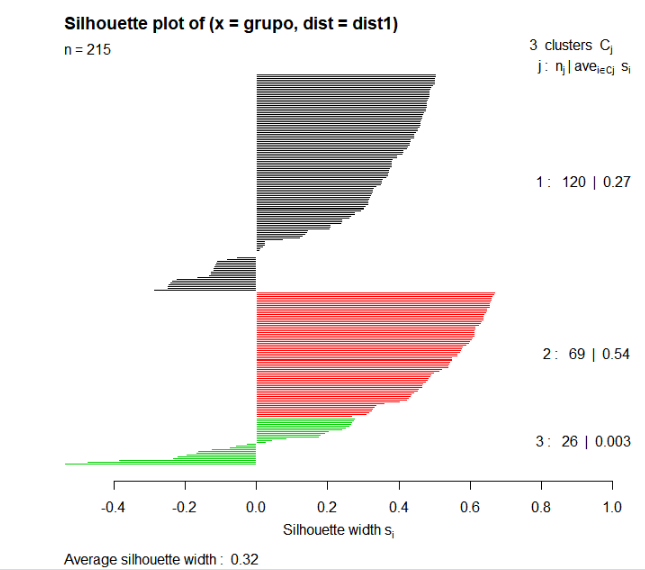


Tiene una media de 0.38, bastante lejana a 1, indicando baja cohesión y separación entre los miembros del clúster.

Al separarlos en 3 clases obtenemos resultados mucho mejores como se observa a continuación:



Sin embargo hay muchos puntos disperses y separados, dando una muy baja cohesión en los clústeres detectados, como se observa en la silueta



El clúster 3 tiene un coeficiente de 0.003, es decir tiene una baja cohesión y gran separación entre sus miembros, mientras que los clústeres 1 y 2 tampoco tiene valores muy elevados.

### Código R ajustado al problema

#Cargo las librerías que voy a utilizar en el código

library(readr)

library(fpc)

library(cluster)

#Cargo el dataset

thyroid <- read\_csv("newthyroid/newthyroid.dat",col\_names = FALSE, col\_types = cols("X6" = col\_factor(levels = c("1","2", "3"))), skip = 10)

names\_thyroid<-scan("newthyroid/newthyroid.dat",what=character(),sep=" ",nlines=11)

names\_thyroid[seq(4,32,by=5)]

names(thyroid)<-names\_thyroid[seq(4,32,by=5)]

thyroid2=thyroid

thyroid2$Class=NULL

#Genero distancia numerico

dist1=dist(thyroid2)

#No se posee distancia binaria para este dataset

#Calculo clúster

h=hclust(dist1,method="ward.D2")

h

plot(h,labels=thyroid$Class)

rect.hclust(h,k=3)

grupo=cutree(h,k=3)

#Analsis de bondad, restrinjo a 150 valores para dibujar

plotcluster(thyroid2,grupo)

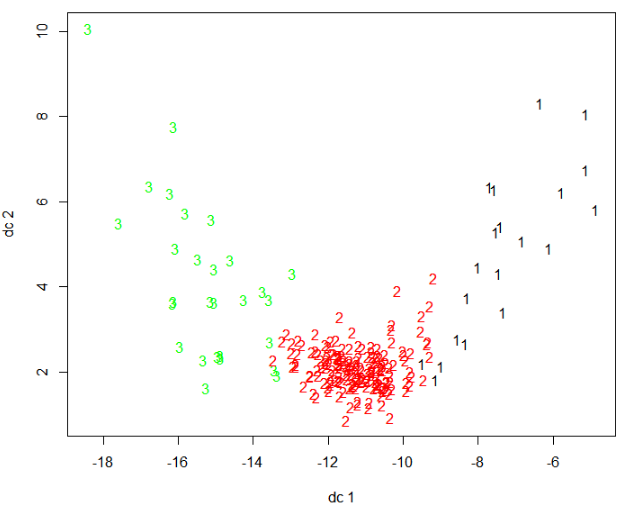
shi= silhouette(grupo,dist1)

plot(shi,col=1:5)

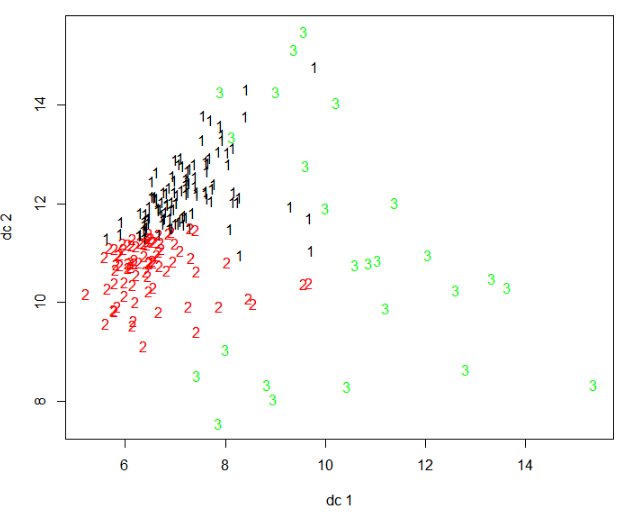
cluster.stats(dist1,grupo)

## Distancias K-medias

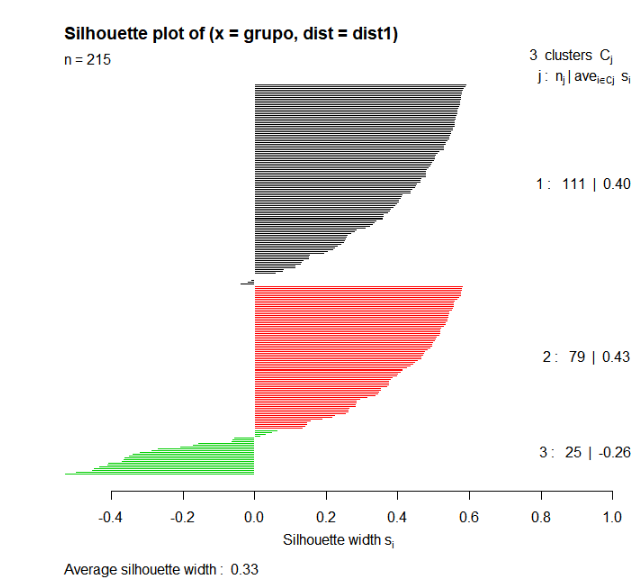
El método de agrupación de las K-medias es un método de agrupación particional, el cual separa todo el conjunto de datos en diferentes clústeres, cuyo número es definido de antemano. El concepto es determinar el número de clústeres que se van a buscar, y los centroides iniciales de dichos clústeres, a partir de estos parámetros iniciales se van reorganizando los elementos pertenecientes a cada clúster sobre varias iteraciones, hasta que convergen a un punto de equilibrio donde no hay más cambios. En el dataset de tiroides sobre el que se trabaja, usando 3 clúster como parámetros de entrada obtenemos la siguiente separación entre los clústeres:



Si bien no está muy mal, tampoco es la mejor agrupación que se podría realizar, esto también está relacionado con el hecho de que los centroides son determinados de forma pseudoaleatoria, lo que nos puede dar resultados diferentes, al volver a realizar el agrupamiento mediante K-medias se puede llegar a obtener una separación como la siguiente:



Bastante más diferente, pero lejos de ideal. Esto se puede observar en la silueta del clúster



El coeficiente de silueta representa la cohesión y separación de los elementos de un clúster y está normalizado y escalado a un rango entre -1 y 1, siendo 1 un clúster ideal, con alta cohesión y cuyos elementos están bien distantes de los elementos de otros clúster.

Aquí se ve que la media del coeficiente es de 0.33 y que el clúster 3 es incluso negativo, mientras que un segmento considerable de los clúster 1 y 2 se encuentran por debajo de 0.3.

### Código R ajustado al problema

#Cargo las librerías que voy a utilizar en el código

library(readr)

library(fpc)

library(cluster)

#Cargo el dataset

thyroid <- read\_csv("newthyroid/newthyroid.dat",col\_names = FALSE, col\_types = cols("X6" = col\_factor(levels = c("1","2", "3"))), skip = 10)

names\_thyroid<-scan("newthyroid/newthyroid.dat",what=character(),sep=" ",nlines=11)

names\_thyroid[seq(4,32,by=5)]

names(thyroid)<-names\_thyroid[seq(4,32,by=5)]

thyroid2=thyroid

thyroid2$Class=NULL

#Genero distancia numerico

dist1=dist(thyroid2)

#No se posee distancia binaria para este dataset

#Calculo Clúster

kmeans.result=kmeans(dist1,3)

kmeans.result$centers

grupo=kmeans.result$cluster

#Analsis de bondad

plotcluster(thyroid2,grupo)

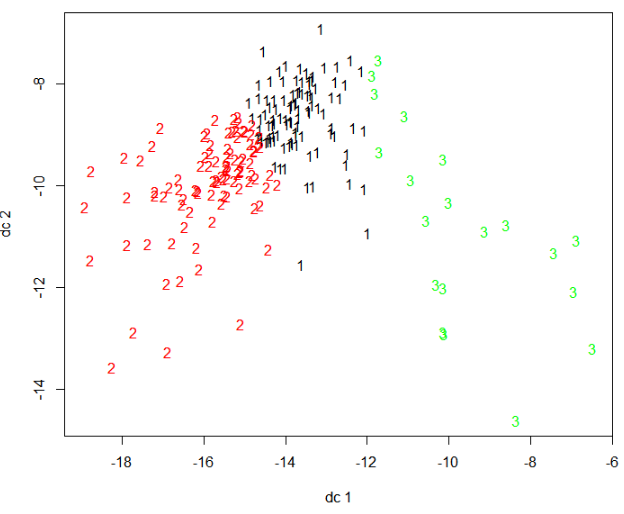
shi= silhouette(grupo,dist1)

plot(shi,col=1:3)

cluster.stats(dist1,grupo)

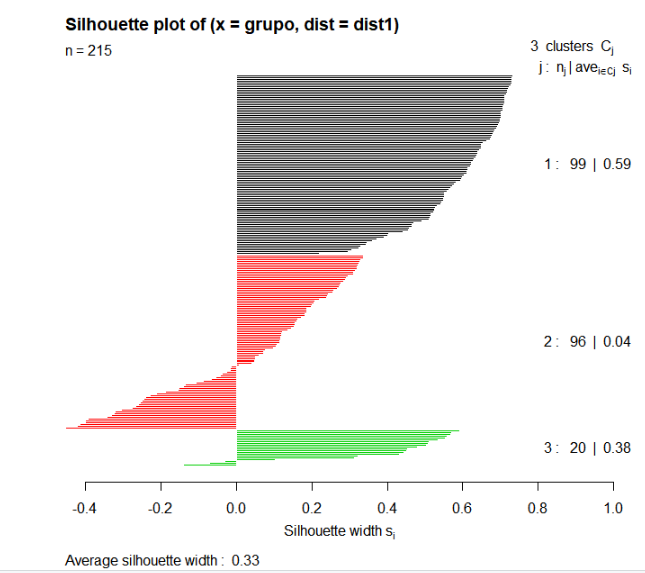
## Distancias K-medoides

El método de los K-medoides es muy parecido al de las K-medias, pero en vez de utilizar la media absoluta, es decir el centroide de cada clúster, se utiliza un punto que sirve como media absoluta, es decir, un medoide. En cada iteración se busca un nuevo punto que sea una mejor representación de la media del clúster. Este proceso se repite hasta que no cambien los medoides, quedando conformados los clúster correspondientes según fue definido en los parámetros de entrada. En el dataset en cuestión, se aplica el método de K-medoides usando la función *pam* en R, lo que da como resultado el siguiente agrupamiento:



A simple vista se ve que es un problema difícil de resolver mediante técnicas de agrupamiento, pero el método de las K-medoides da un resultado aceptable.

La silueta de los clústeres, para hacer mediciones de bondad es la siguiente:



En este caso observamos que el clúster 1 es bastante decente, con un coeficiente de 0.59, pero el clúster número 2, que contiene solo 3 elementos menos que el clúster principal tiene un coeficiente de 0.04, con elementos con coeficiente inferior a -0.4, indicando mucha separación y poca cohesión entre sus elementos. El tercer clúster a su vez dista de ser ideal, con su punto más elevado alcanzando solo un valor de 0.6 y el resto decayendo rápidamente.

### Código R ajustado al problema

#Cargo las librerías que voy a utilizar en el código

library(readr)

library(fpc)

library(cluster)

#Cargo el dataset

thyroid <- read\_csv("newthyroid/newthyroid.dat",col\_names = FALSE, col\_types = cols("X6" = col\_factor(levels = c("1","2", "3"))), skip = 10)

names\_thyroid<-scan("newthyroid/newthyroid.dat",what=character(),sep=" ",nlines=11)

names\_thyroid[seq(4,32,by=5)]

names(thyroid)<-names\_thyroid[seq(4,32,by=5)]

thyroid2=thyroid

thyroid2$Class=NULL

#Genero distancia numerico

dist1=dist(thyroid2)

#No se posee distancia binaria para este dataset

#Calculo Clúster

pam.result=pam(dist1,3)

grupo=pam.result$clustering

#Analsis de bondad

plotcluster(thyroid2,grupo)

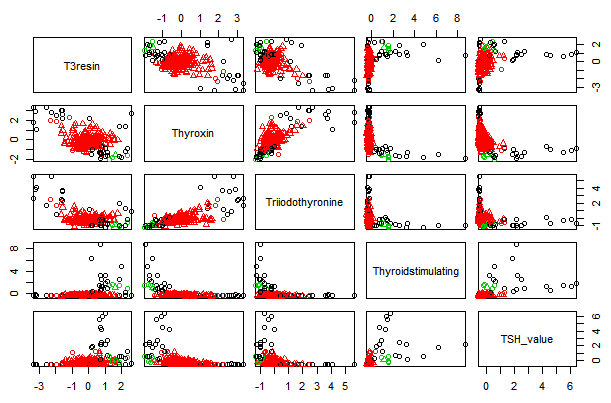
shi= silhouette(grupo,dist1)

plot(shi,col=1:3)

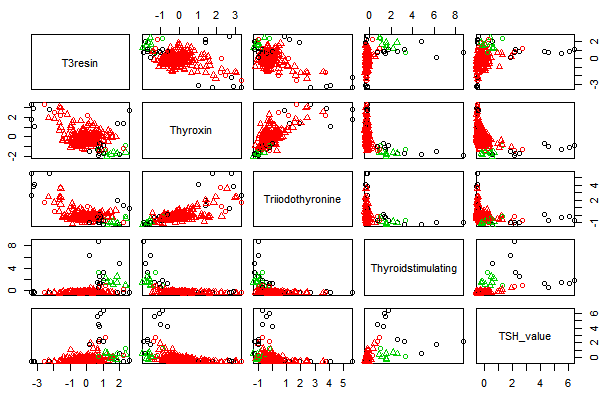
cluster.stats(dist1,grupo)

## DBScan

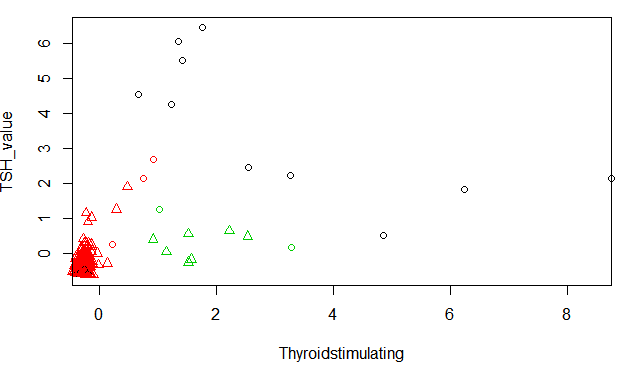
DBScan es un método de agrupamiento particional, el cual separa los diferentes los diferentes clústeres basado en la detección de puntos núcleos, los puntos enlazados al mismo que conforman el clúster, los puntos en la frontera de los mismos y los puntos ruido que no pertenecen a ningún clúster, estos puntos también son considerados un clúster desde un punto de vista analítico, pero la distancia entre dichos puntos hace que no sea práctico considerarlos un clúster para el problema en cuestión en la mayoría de los casos. DBScan tiene dos parámetros, la distancia entre puntos para enlazarlos al clúster y el mínimo de puntos que deben encontrarse a menor de esa distancia de un punto para considerar a este punto un punto núcleo y por ende, un clúster. Como la gran mayoría de los algoritmos de agrupamiento no supervisados, el DBScan tiene una gran variación en su resultado basado en estos parámetros. En el dataset en cuestión se puede observar la siguiente agrupación si se utiliza una distancia de 1 y un mínimo de puntos de 5:



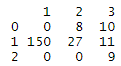
Básicamente agrupa una gran parte del dataset en un clúster principal, un pequeño clúster secundario y el resto de puntos muy lejanos que son considerados ruido. Si aumentamos un poco la distancia de agrupamiento a 1.5 se pueden obtener algunos grupos un poco más notables como se puede observar:



Si observamos con más cuidado la combinación de TSH\_value con Thyroidstimulating se puede observar la separación entre un gran clúster principal muy denso y un pequeño grupo señalado con el color verde.

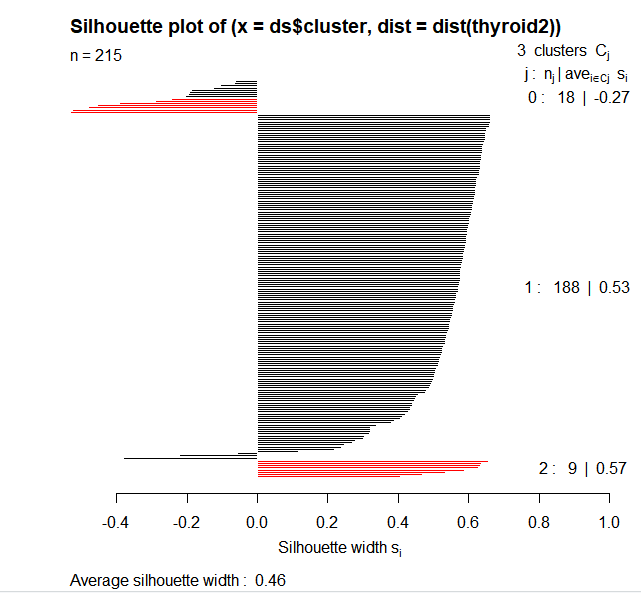


En este caso se puede decir que DBScan hizo una agrupación más o menos aceptable, sin embargo, en este dataset se poseen los valores reales de las clases a las que pertenecen cada caso y se puede hacer una comparación la cual nos da la siguiente tabla:



Se puede observar que DBScan agrupa correctamente los 150 casos de la clase 1 en un solo cluster, pero también agrupa 27 de la clase 2 y 11 de la clase 3 como parte del mismo cluster, así que en realidad no tiene una precisión muy elevada en este dataset con las parámetros establecidos.

La silueta de los grupos formados es la siguiente:



La silueta combina en una sola gráfica las medidas de separación y cohesión, con una variación entre -1 y 1, siendo 1 un clúster denso, con puntos cercanos entre sí, y bien distanciados del resto de los puntos no pertenecientes al clúster. Como se puede observar, el cluster principal con 188 elementos tiene una silueta de solo 0.53, indicando que los puntos no están muy agrupados y que la distancia que los separa de otros puntos fuera del clúster no es muy elevada, lo mismo pasa con el pequeño clúster secundario con una medida de 0.57, mientras que el clúster formado por los puntos ruido tiene un coeficiente negativo, indicando que efectivamente están totalmente dispersos y más cercanos a otros clústeres que a ellos mismos.

### Código R ajustado al problema

#Cargo las librerías que voy a utilizar en el código

library(readr)

library(fpc)

library(cluster)

#Cargo el dataset

thyroid <- read\_csv("newthyroid/newthyroid.dat",col\_names = FALSE, col\_types = cols("X6" = col\_factor(levels = c("1","2", "3"))), skip = 10)

names\_thyroid<-scan("newthyroid/newthyroid.dat",what=character(),sep=" ",nlines=11)

names\_thyroid[seq(4,32,by=5)]

names(thyroid)<-names\_thyroid[seq(4,32,by=5)]

thyroid2=thyroid

thyroid2$Class=NULL

#normalizo los datos para DBScan

for (j in 1:5) {

x=thyroid2[[j]];

v=(x-mean(x))/sqrt(var(x));

thyroid2[[j]]=v

}

#Aplico DBscan

ds=dbscan(thyroid2,eps=1.5,MinPts=5)

#Tabla de comparación

table(ds$cluster,thyroid$Class)

#Gráficas

plot(ds,thyroid2)

plot(ds,thyroid2[c(4,5)])

#Silueta para determinar cohesión y separación

x=table(ds$cluster,thyroid$Class)

nc=length(x[,1])

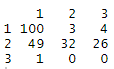
shi= silhouette(ds$cluster,dist(thyroid2))

plot(shi,col=1:nc)

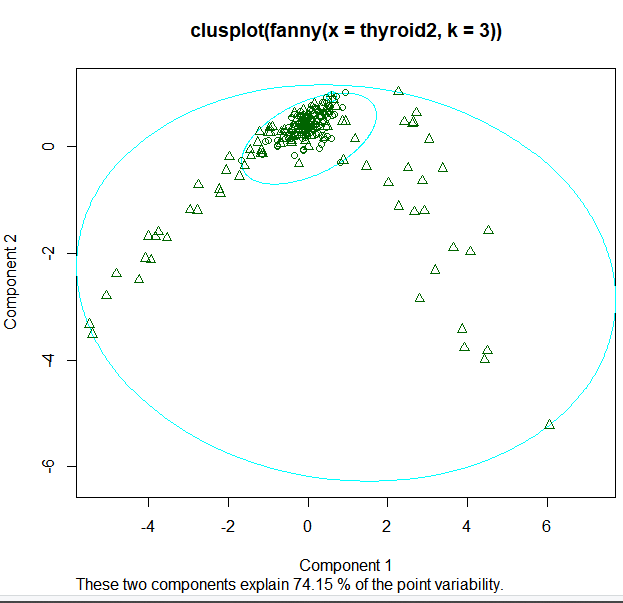
## K-medias difuso normalizado

La lógica de todo algoritmo de agrupamiento difuso está basado en la pertenencia de un punto a los diferentes clúster, típicamente usado cuando el conjunto de datos tiene etiquetas imprecisas o difusas como la gama de un coche o la calidad de una empresa. Basado en esto, los algoritmos desarrollan funciones de pertenencia y no de probabilidad, en vez de definir qué A es de la clase x con 80% de certeza, se define qué A posee un 80% de las características que lo definirían como una clase X, esto no descarta la posibilidad de que A pertenezca a otra clase y que simplemente posea características de x, esto es muy útil cuando se tiene grupos que se solapan y entremezclan, haciendo que sea difícil separar las barreras entre grupos, ya que estos poseen zonas comunes, típicamente en múltiples dimensiones.

En R poseemos la función *Fanny* que permite generar los conjuntos difusos del dataset a partir de un grupo de parámetros. Si se aplica esta función al dataset Thyroid con la intención de detectar 3 clústeres se obtiene lo siguiente:

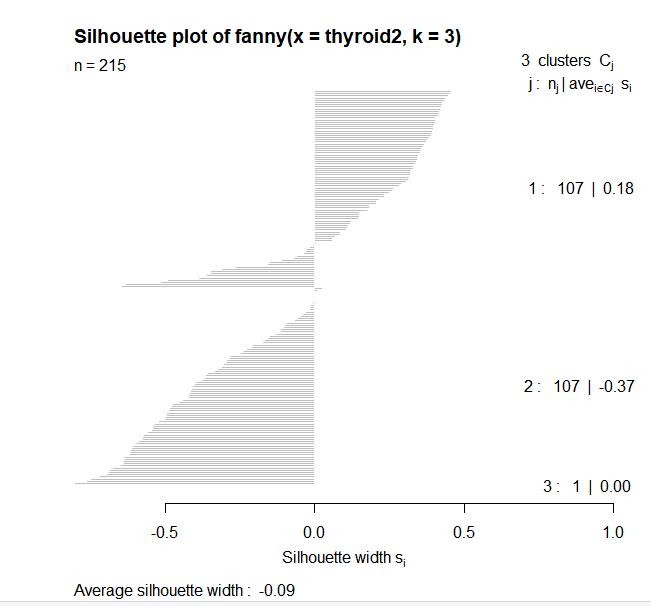


En este caso vemos que se agrupan como miembros de la clase 2 a muchos de los miembros de la clase 1, lo cual impacta significativamente su precisión, lo cual es comprensible si se analiza las gráficas generadas:



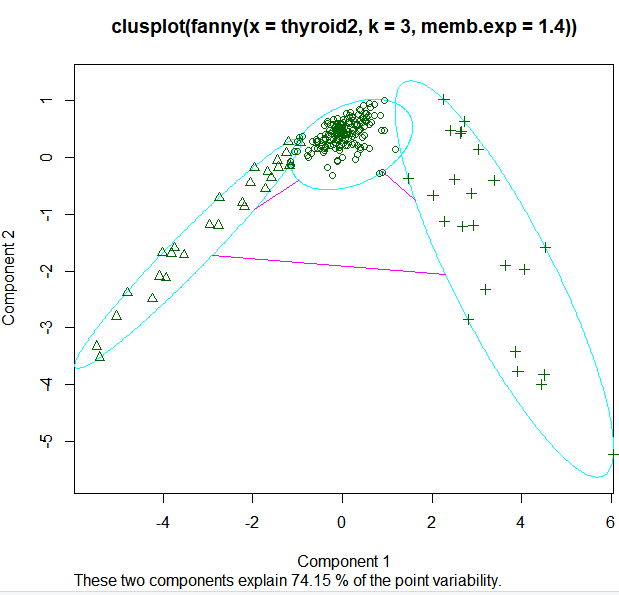
Se observa como casi todos los ejemplos están muy densos y concentrados en una zona, haciendo muy difícil su separación correcta en diferentes clases.

La silueta resultante de este análisis es la siguiente

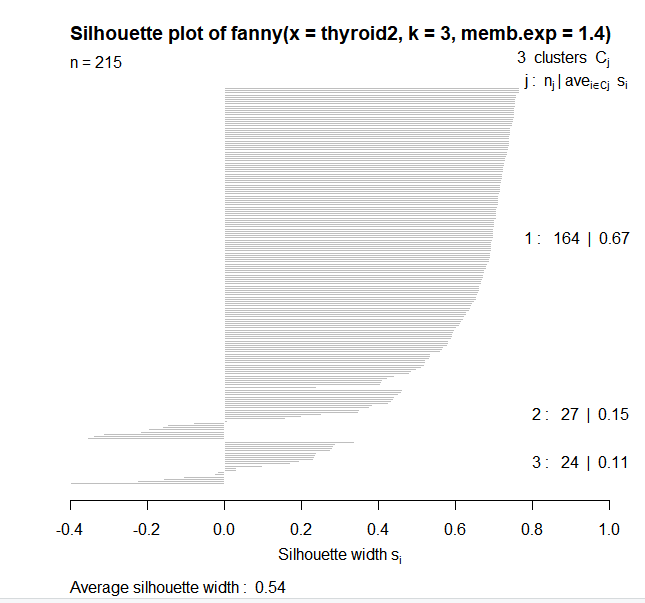


Esta silueta posee una media de -0.09, indicando una muy mala separación y cohesión entre los clúster resultantes lo cual indica que probablemente no estemos en presencia de una buena separación entre los clúster.

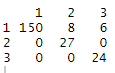
Sin embargo, como todos los algoritmos de agrupamiento, un cambio en los parámetros de entrada puede dar resultados drásticamente diferentes. Al modificar el exponente de pertenencia de *Fanny* y reducirlo a 1.4, se observan 3 clusteres mucho más separados que antes:



Una silueta con una media de 0.54



Y una calificación perfecta significativamente mejor en los clústeres generados:



Cometiendo solo 14 errores, para una precisión del 93.4%.

### Código R ajustado al problema

#Cargo las librerías que voy a utilizar en el código

library(readr)

library(fpc)

library(cluster)

#Cargo el dataset

thyroid <- read\_csv("newthyroid/newthyroid.dat",col\_names = FALSE, col\_types = cols("X6" = col\_factor(levels = c("1","2", "3"))), skip = 10)

names\_thyroid<-scan("newthyroid/newthyroid.dat",what=character(),sep=" ",nlines=11)

names\_thyroid[seq(4,32,by=5)]

names(thyroid)<-names\_thyroid[seq(4,32,by=5)]

thyroid2=thyroid

thyroid2$Class=NULL

#normalizo los datos

for (j in 1:5) {

x=thyroid2[[j]];

v=(x-mean(x))/sqrt(var(x));

thyroid2[[j]]=v

}

#Trabajo con clustering difuso

fuzzy.result=fanny(thyroid2,3,memb.exp=1.4)

str(fuzzy.result)

fuzzy.result$membership

table(fuzzy.result$clustering,thyroid$Class)

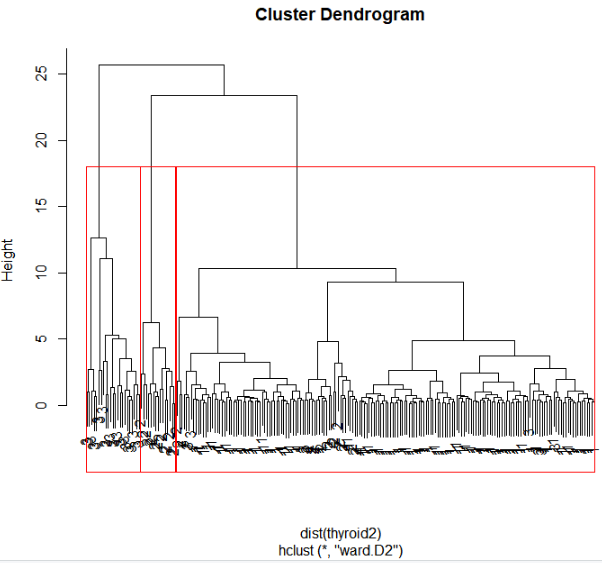
plot(fuzzy.result)

#(datos adicionales de la calidad del agrupamiento realizado)

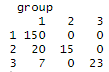
cluster.stats(dist(thyroid),fuzzy.result$clustering,alt.clustering=as.integer(thyroid$Class))

## Jerárquico normalizado

Este método es simplemente una variación del método de distancias jerárquico, con un paso adicional para normalizar los datos. No obstante, la normalización de los datos puede causar un gran impacto en la generación de la jerarquía como se observa a continuación:

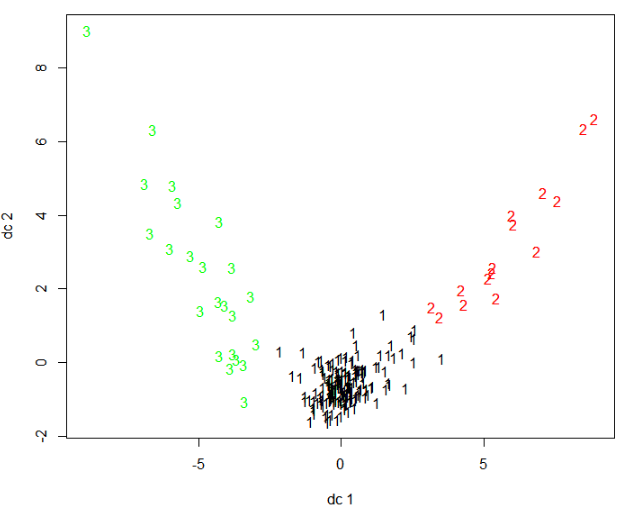


Es importante resaltar como la altura del dendograma fue reducida de 215 a 25, y que los clústeres resultantes tienen una calidad mejor en general que para el caso no normalizado.

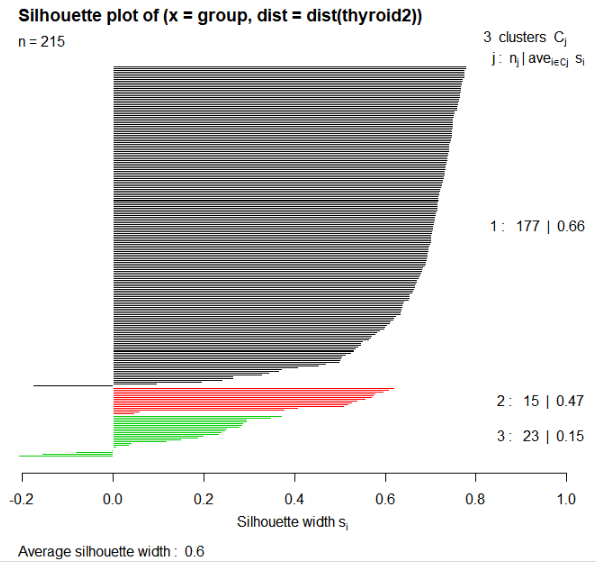


Califica 20 casos de hipertiroidismo como normal, y 7 de hipertiroidismo como normal, efectivamente incluyendo 27 casos incorrectos en el clúster de personas normales, lo cual nos da una precisión de 87.4%.

Las medidas de bondad del clúster normalizado es también mejor, como se observa en la siguiente imagen, tenemos los clústeres más separados entre ellos, sin solapamiento alguno.



La silueta de los clústeres resultantes también da resultados bastante buenos, aunque como se puede observar, los clústeres 2 y 3 tienen bastante separación entre sus miembros, lo que reduce el valor del coeficiente general a 0.6



### Código R ajustado al problema

#Cargo las librerías que voy a utilizar en el código

library(readr)

library(fpc)

library(cluster)

#Cargo el dataset

thyroid <- read\_csv("newthyroid/newthyroid.dat",col\_names = FALSE, col\_types = cols("X6" = col\_factor(levels = c("1","2", "3"))), skip = 10)

names\_thyroid<-scan("newthyroid/newthyroid.dat",what=character(),sep=" ",nlines=11)

names\_thyroid[seq(4,32,by=5)]

names(thyroid)<-names\_thyroid[seq(4,32,by=5)]

thyroid2=thyroid

thyroid2$Class=NULL

#normalizo los datos

for (j in 1:5) {

x=thyroid2[[j]];

v=(x-mean(x))/sqrt(var(x));

thyroid2[[j]]=v

}

#calculo cluster jeraquico por el metodo de Ward, distancia euclidea

hc=hclust(dist(thyroid2),method="ward.D2")

hc

#dibujo el dendrograma y corto por tres

plot(hc,labels=thyroid$Class)

rect.hclust(hc,k=3)

#genero la variable de agrupamiento

group=cutree(hc,k=3)

group

table(thyroid$Class,group)

#Medidas de bondad de agrupamiento: coeficiente de silueta

plotcluster(thyroid2,group)

shi= silhouette(group,dist(thyroid2))

plot(shi,col=1:3)

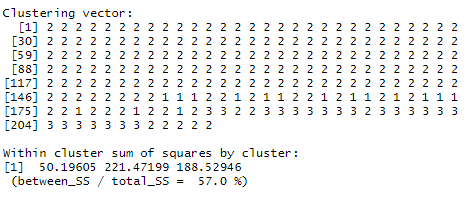
#Calculo de algunas otras medidas de bondad del agrupamiento

cluster.stats(dist(thyroid2),group,alt.clustering=as.integer(thyroid$Class))

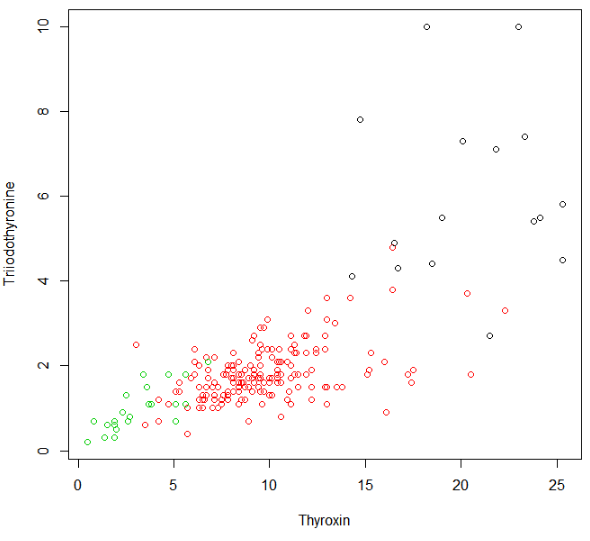
## K-medias normalizado

Este método es simplemente una variación del método de K-medias, con un paso adicional para normalizar los datos. Similar al caso anterior, algo tan simple como un proceso de escalado y normalización puede influir enormemente en el resultado obtenido.

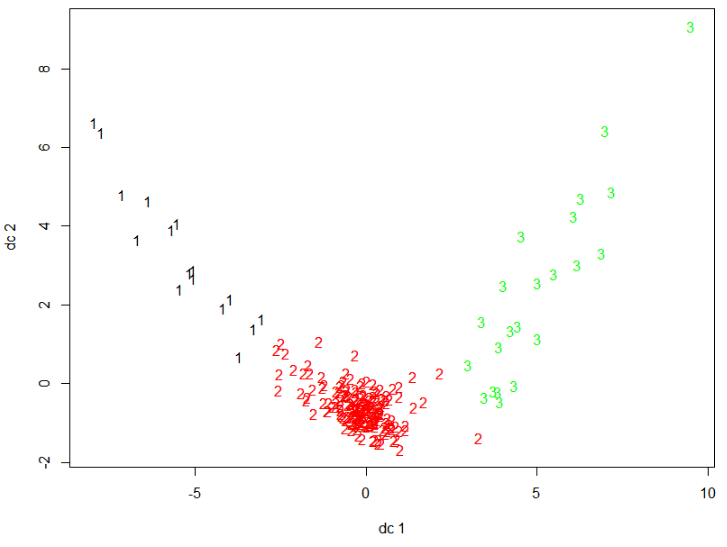
En este caso tenemos la siguiente asignación de vectores:



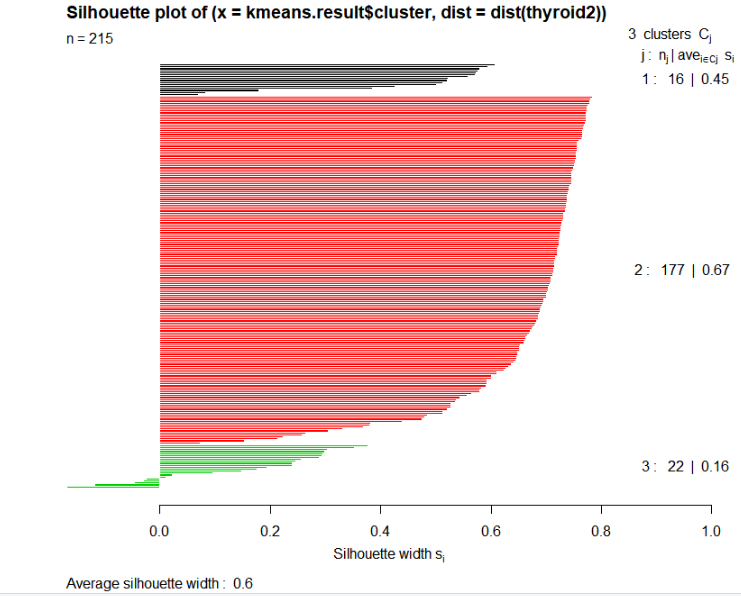
Además, se puede observar que las distancias dentro de los clústeres con respecto al total es de 57%, típicamente un porcentaje menor indica que los clústeres están más compactos, en este caso tenemos que si hacemos una representación gráfica nos quedamos con:



El cual cómo podemos observar está bastante disperso y tiene bastantes puntos entremezclados, si hacemos una representación del clúster usando la distancia en vez de la representación en el conjunto de datos obtenemos la siguiente agrupación:



Que es mucho mejor a la anterior, de hecho una medición de bondad con la silueta nos da un valor promedio de 0.6, mostrando que tenemos un grupo de clústeres bastante cohesivos y separados, no ideal, pero aceptable.



No obstante, a pesar de tener buena calidad los clústeres detectados, la precisión no es ideal, y la función *cluster.stats* que nos da un conjunto de métricas de calidad de los clústeres nos da un índice de variación de información muy elevado, 0.6210801, es decir, un 62.1% de variación entre los clústeres generados y los valores originales contenidos en el dataset, lo cual indica una pésima precisión a la hora de asignar los puntos a los clústeres, generando una gran variación de información.

### Código R ajustado al problema

#Cargo las librerías que voy a utilizar en el código

library(readr)

library(fpc)

library(cluster)

#Cargo el dataset

thyroid <- read\_csv("newthyroid/newthyroid.dat",col\_names = FALSE, col\_types = cols("X6" = col\_factor(levels = c("1","2", "3"))), skip = 10)

names\_thyroid<-scan("newthyroid/newthyroid.dat",what=character(),sep=" ",nlines=11)

names\_thyroid[seq(4,32,by=5)]

names(thyroid)<-names\_thyroid[seq(4,32,by=5)]

thyroid2=thyroid

thyroid2$Class=NULL

#normalizo los datos

for (j in 1:5) {

x=thyroid2[[j]];

v=(x-mean(x))/sqrt(var(x));

thyroid2[[j]]=v

}

#Aplico K-means

kmeans.result=kmeans(thyroid2,3)

kmeans.result

#Tabla de comparación

table(thyroid$Class,kmeans.result$cluster)

#Analisis de bondad

plot(thyroid[2:3], col=kmeans.result$cluster)

points(kmeans.result$centers[2:3],col=1:3,pch=8,cex=2)

x=kmeans.result$cluster

plotcluster(thyroid2,x)

shi= silhouette(kmeans.result$cluster,dist(thyroid2))

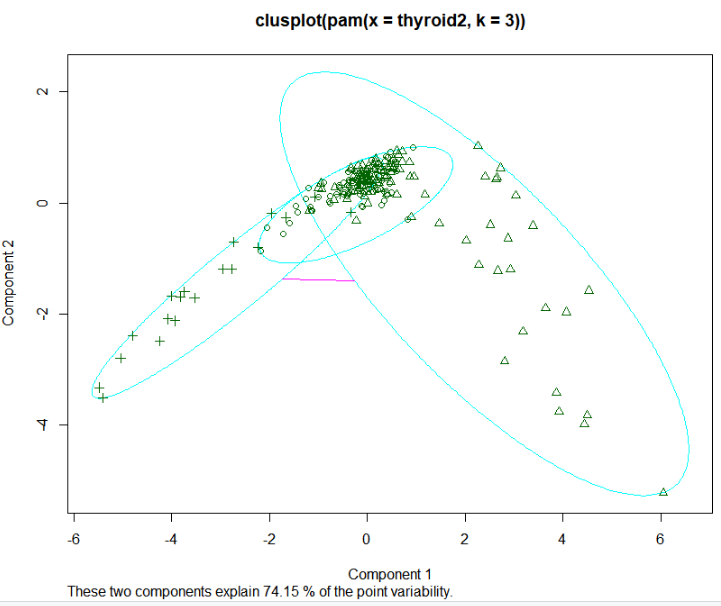
plot(shi,col=1:3)

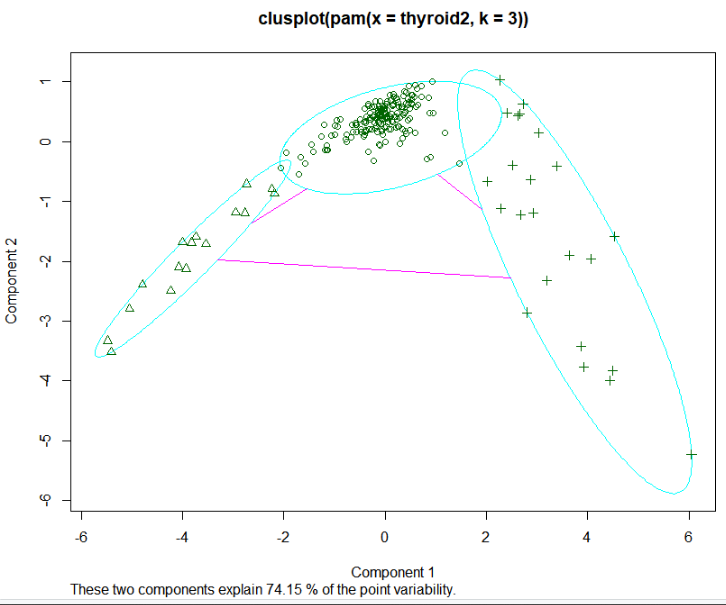
#Calculo de algunas otras medidas de bondad del agrupamiento

group=kmeans.result$cluster

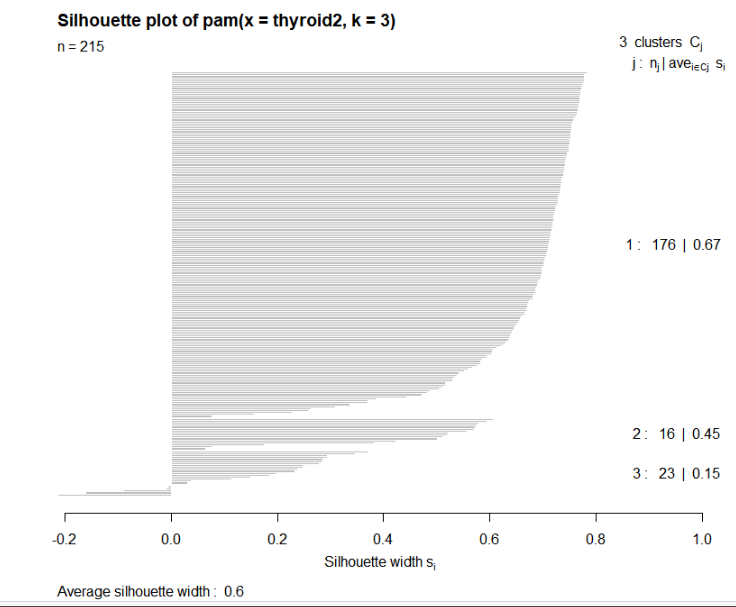
cluster.stats(dist(thyroid2),group,alt.clustering=as.integer(thyroid$Class))

## Medoides normalizado

Este método es simplemente una variación del método de K-medoides, con un paso adicional para normalizar los datos. Como en los casos anteriores, la normalización de los parámetros de entrada puede afectar significativamente los resultados del algoritmo, como se puede ver a continuación: 

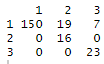


La primera imagen muestra la agrupación con los datos naturales, mientras que la segunda muestra la agrupación con los datos normalizados. Es fácil observar que no existe mucha mezcla entre los clústeres en la agrupación realizada con los datos normalizados.



La silueta muestra un coeficiente bastante elevado para el clúster principal, y una media de 0.6, indicando una cohesión y separación aceptable, aunque no ideal.

La precisión es relativamente baja, ya que tenemos 19 personas con hipertiroidismo y 7 con hipotiroidismos clasificados como personas normales, lo cual es muy malo, puesto que no recibirían el tratamiento adecuado:



### Código R ajustado al problema

#Cargo las librerías que voy a utilizar en el código

library(readr)

library(fpc)

library(cluster)

#Cargo el dataset

thyroid <- read\_csv("newthyroid/newthyroid.dat",col\_names = FALSE, col\_types = cols("X6" = col\_factor(levels = c("1","2", "3"))), skip = 10)

names\_thyroid<-scan("newthyroid/newthyroid.dat",what=character(),sep=" ",nlines=11)

names\_thyroid[seq(4,32,by=5)]

names(thyroid)<-names\_thyroid[seq(4,32,by=5)]

thyroid2=thyroid

thyroid2$Class=NULL

#normalizo los datos

for (j in 1:5) {

x=thyroid2[[j]];

v=(x-mean(x))/sqrt(var(x));

thyroid2[[j]]=v

}

#K-medoides clasico (pam) con 3 clases.

pam.result=pam(thyroid2,3)

table(pam.result$clustering,thyroid$Class)

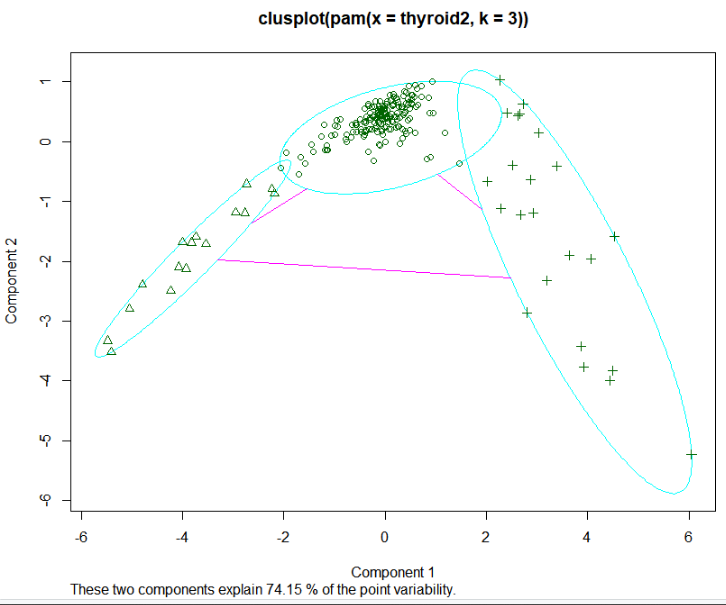
plot(pam.result)

group=pam.result$clustering

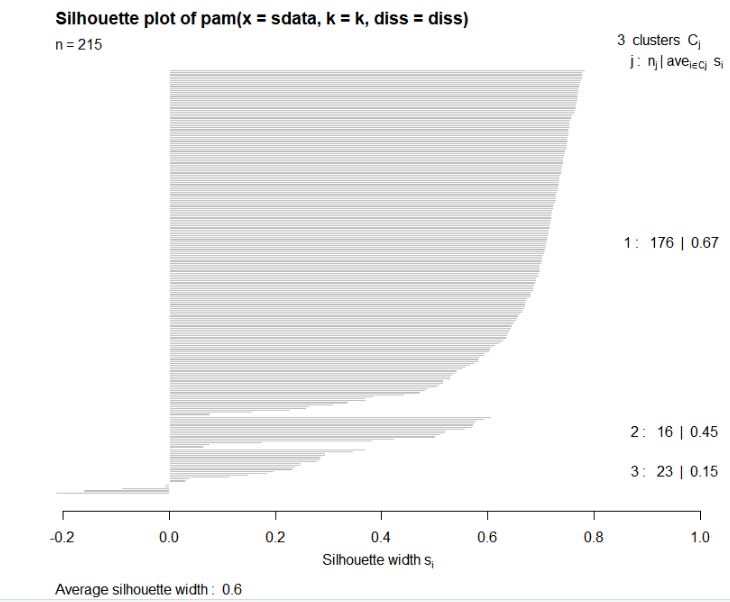
cluster.stats(dist(thyroid2),group,alt.clustering=as.integer(thyroid$Class))

## Medoides normalizado con valor óptimo de grupos

Este método es simplemente una variación del método de K-medoides normalizado, utilizando la función *pamk* en vez de *pam* para la detección del número óptimo de grupos. *Pamk* determina automáticamente una distribución óptima de clústeres en el espacio, a cambio de la necesidad de utilizar más poder de procesamiento para hacer los cálculos. En el conjunto de datos, como el número inicial de clases eran 3, *pamk* determino la separación en 3 clústeres, lo cual nos da el mismo valor que en K-medoides normalizado:



Y la misma silueta:



No hay mucho más que explicar excepto que la determinación de grupos óptima puede ser útil cuando se tiene un dataset relativamente pequeño del cual no se tiene absolutamente ninguna información y se quiere establecer algún tipo de agrupamiento y separación, pero no se conoce a priori cuantos clústeres deben ser creados.

El resultado de *pamk* es simplemente un *pam* con el número óptimo de clústeres resultante como parámetro de entrada.

### Código R ajustado al problema

#Cargo las librerías que voy a utilizar en el código

library(readr)

library(fpc)

library(cluster)

#Cargo el dataset

thyroid <- read\_csv("newthyroid/newthyroid.dat",col\_names = FALSE, col\_types = cols("X6" = col\_factor(levels = c("1","2", "3"))), skip = 10)

names\_thyroid<-scan("newthyroid/newthyroid.dat",what=character(),sep=" ",nlines=11)

names\_thyroid[seq(4,32,by=5)]

names(thyroid)<-names\_thyroid[seq(4,32,by=5)]

thyroid2=thyroid

thyroid2$Class=NULL

#normalizo los datos

for (j in 1:5) {

x=thyroid2[[j]];

v=(x-mean(x))/sqrt(var(x));

thyroid2[[j]]=v

}

#Pruebo con k-medoides optimo numero de grupos

pamk.result=pamk(thyroid2)

pamk.result

pamk.result$nc

table(as.vector(pamk.result$pamobject$clustering),thyroid$Class)

plot(pamk.result$pamobject)

#(Ver descripcion de la funcion)

group=pamk.result$pamobject$clustering

cluster.stats(dist(thyroid2),group)