Laboratorio Clustering

Fernando Crema

March 17, 2016

kmedias

Empezaremos con el algoritmo kmedias

Lectura

Primero leemos los datos asumiendo que se hizo `setwd(directorio)` del directorio en dónde estén los datos del laboratorio.

```
datos = read.csv(file = "entrada_1.csv",
                 header = T,
                 row.names = 1)
```

Analizamos los datos dentro del dataset

```
head(datos)
```

```
##            X        Y class
## 1  5.0402321 3.500179     1
## 2 11.0270276 3.818685     1
## 3  7.9292809 4.701487     1
## 4  5.4389327 4.353904     1
## 5  0.5046812 4.334847     1
## 6 17.2996780 4.583857     1
```

Colocamos la columna class para poder analizar los algoritmos de clustering

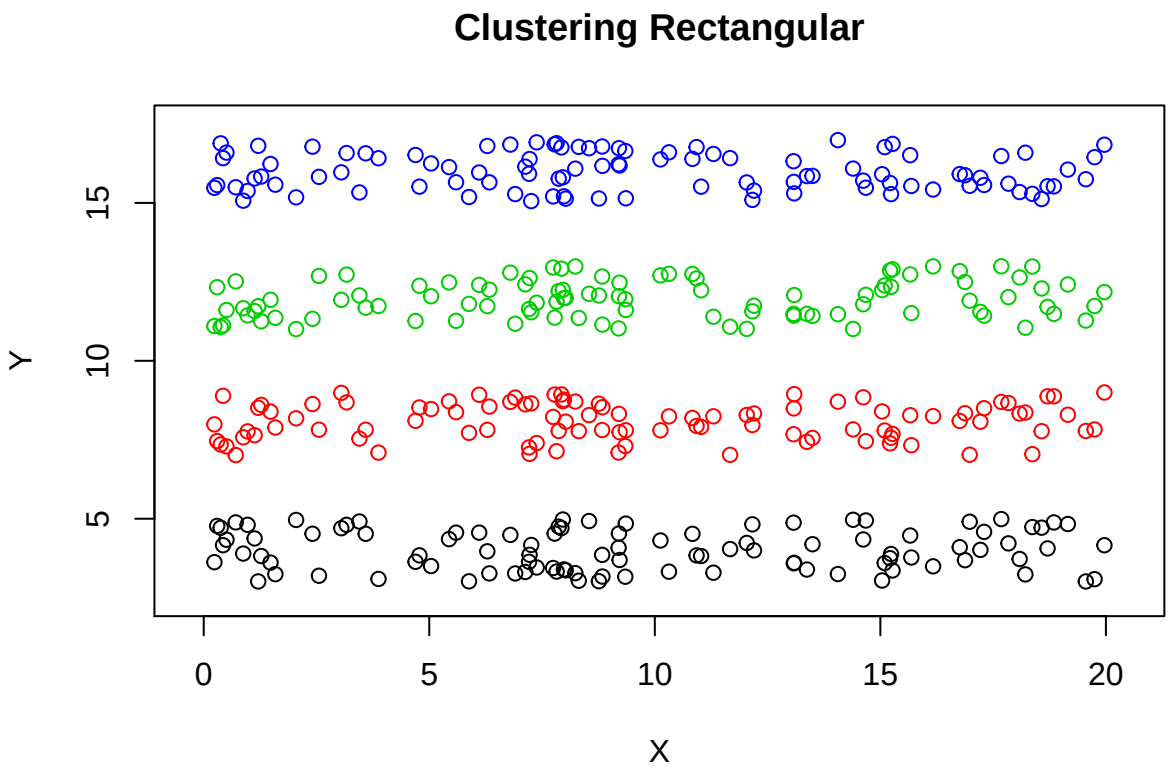
En el caso del dataset nro1 tenemos 100 individuos en cada clase.

```
table(datos$class)
```

```
##
##   1   2   3   4
## 100 100 100 100
```

Graficamos el dataset 1 con colores para cada clase

```
plot(datos$X,
     datos$Y,
     col = datos$class,
     xlim = c(min(datos$X-0.5), max(datos$X+0.5)),
     ylim = c(min(datos$Y-0.5), max(datos$Y+0.5)),
     xlab = "X",
     ylab = "Y",
     main = "Clustering Rectangular")
```



Usando el algoritmo kmeans del paquete stats

Hallando el modelo

Para llamar al método necesitamos usar solamente las columnas X y Y porque las clases no las debería recibir el método.

```
modelo.k.medias = kmeans(x = datos[, c("X", "Y")],
                         centers = 4)
```

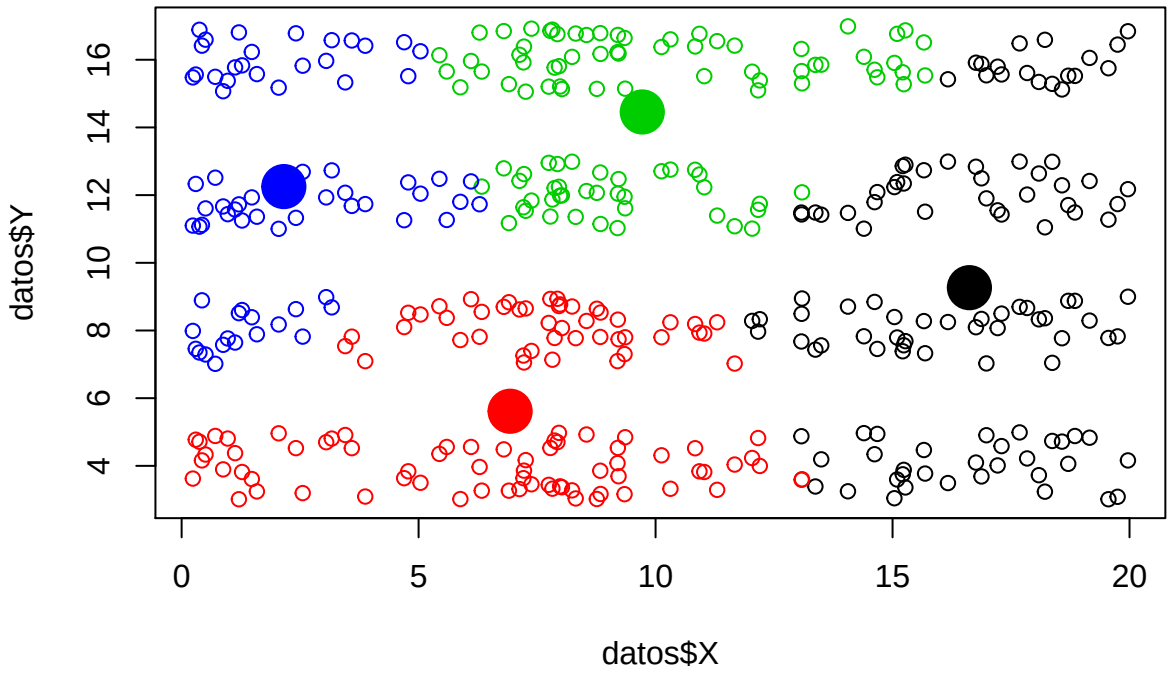
Graficando el modelo

Para graficar el modelo necesitamos:

- Graficar datos originales.
- Los colores de los datos serán los identificadores de los cluster que tenemos en `cluster`.
- Graficar los centroides.

```
# Datos originales con colores de cluster
plot(x = datos$X,
     y = datos$Y,
     col = modelo.k.medias$cluster)
```

```r
# Ahora graficamos los puntos
points(x = modelo.k.medias$centers[, c("X", "Y")],
       col = 1:4, pch = 19, cex = 3)
```



Es posible que dos ejecuciones distintas de kmedias bajo las mismas condiciones den clusters y centros distintos. Esto se debe a que si solo especificamos el número de cluster entonces el método decide centroides aleatorios.

Rendimiento del modelo

Si quisiéramos ver el rendimiento de kmedias en el dataset 1 podemos hacer la matriz de confusión usando la columna class del dataset contra los cluster de salida del método.

Nota: Es posible que los colores de los cluster y las clases estén cambiados y pareciera que la matriz no tiene sentido. Hay que tener cuidado al analizar esto.

```r
table(modelo.k.medias$cluster, datos$class)
```

```
##
##      1  2  3  4
##   1 32 37 33 18
##   2 68 45  0  0
##   3  0  0 38 58
##   4  0 18 29 24
```

Usando los demás datos

En vista de que tenemos 4 datasets distintos y pasos son iguales para cada dataset entonces hacemos una función para reutilizarla en cada dataset.

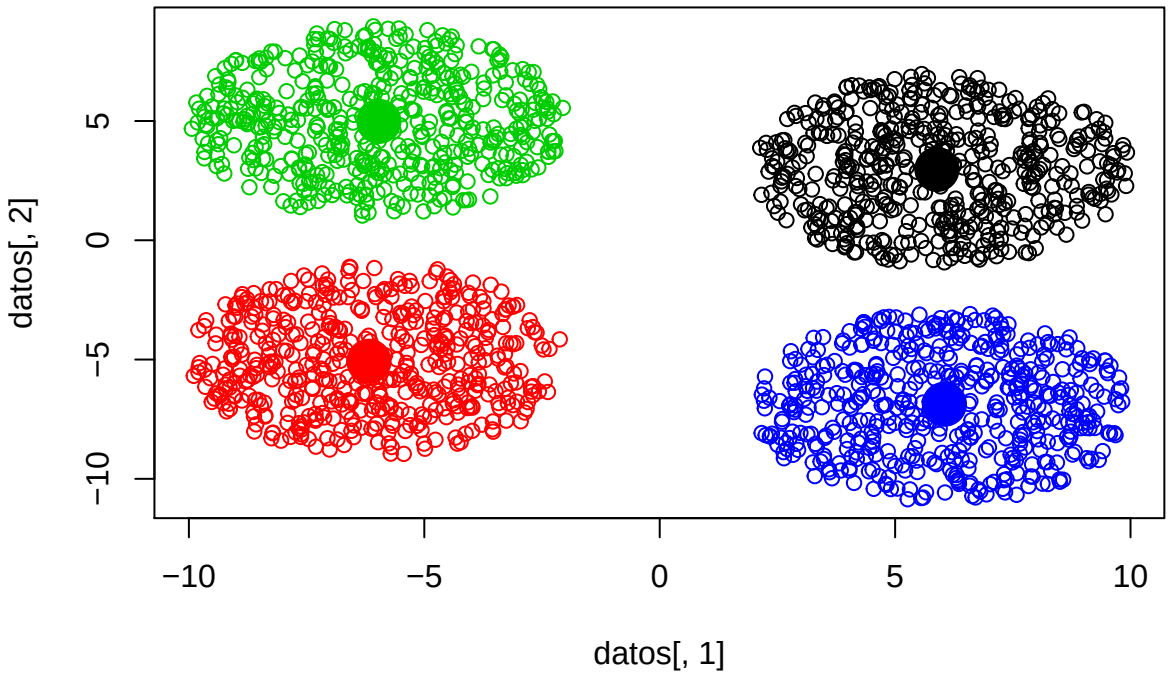
Por ahora, debería recibir:

- Nombre del archivo de entrada.
- El número de clusters que quiero analizar.

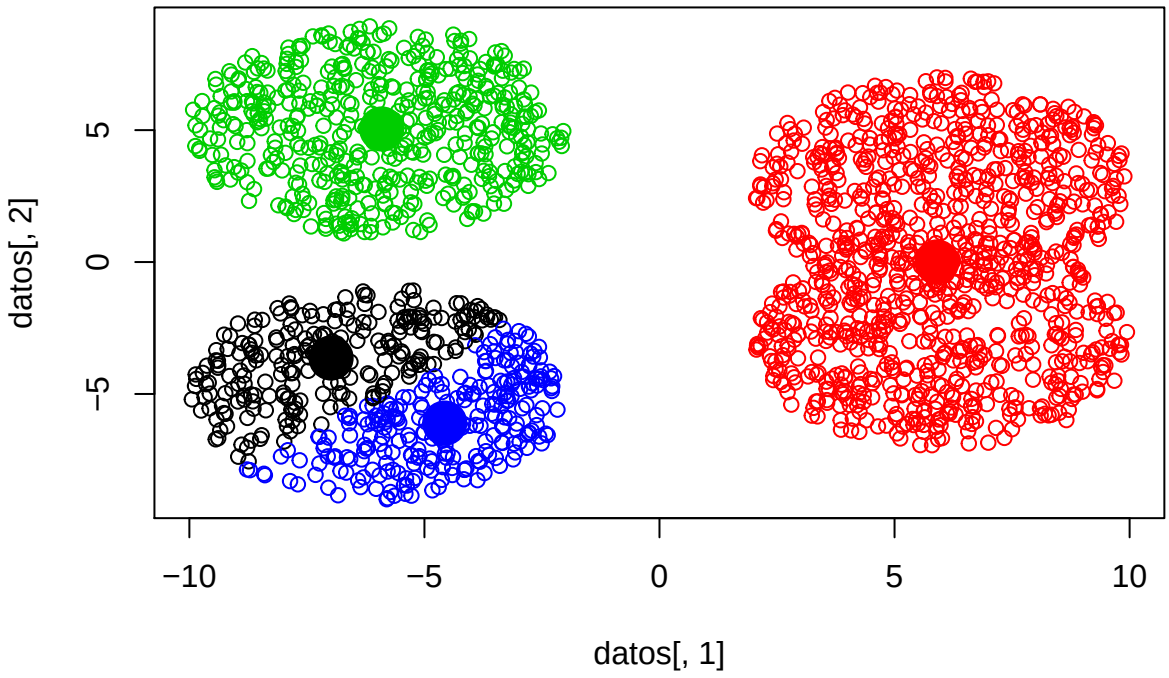
```
kmedias = function(archivo, k){
  # Abrimos archivo
  datos = read.csv(file = archivo,
                   header = T,
                   row.names = 1)

  # Encontramos modelo
  modelo.k.medias = kmeans(x = datos[, 1:2],
                           centers = k)

  # Datos originales con colores de cluster
  plot(x = datos[, 1],
       y = datos[, 2],
       col = modelo.k.medias$cluster)

  # Ahora graficamos los puntos
  points(x = modelo.k.medias$centers[, 1:2],
         col = 1:4, pch = 19, cex = 3)
}
```

Usamos ahora la función para cada dataset.

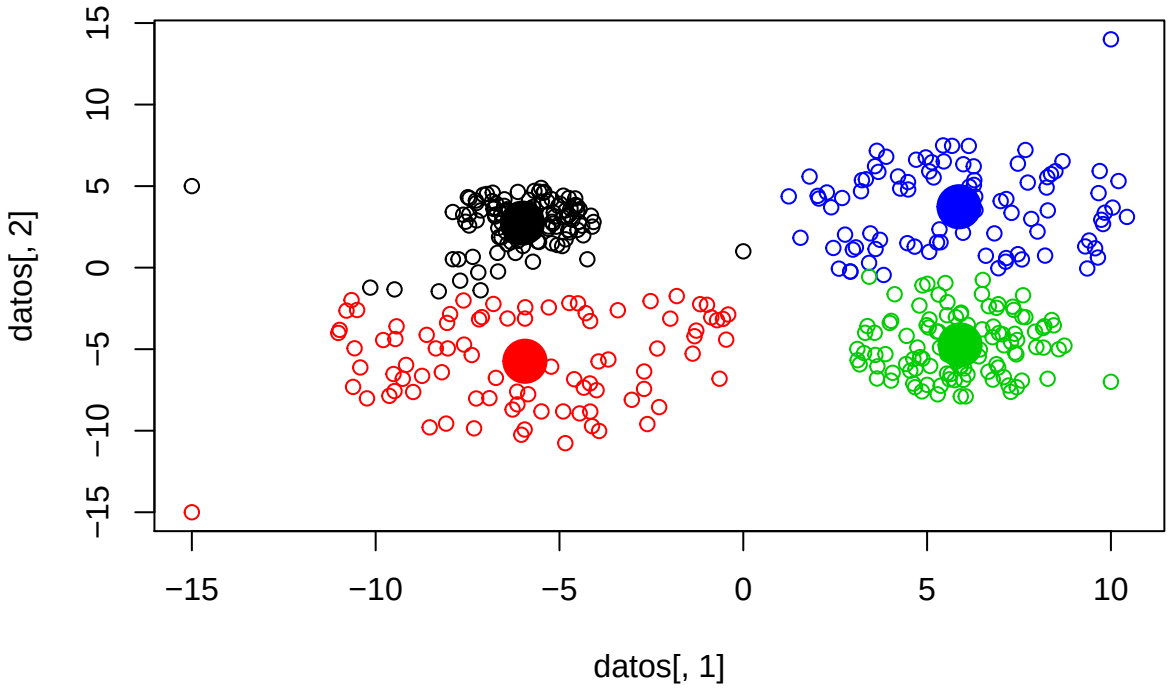
```
  kmedias("entrada_2.csv", 4)
```




```
kmedias("entrada_3.csv", 4)
```




```
kmedias("entrada_4.csv", 4)
```



Cluster Jerárquicos

La ventaja de usar algoritmos de clúster jerárquicos con respecto a kmedias, radica en el hecho de reutilizar la estructura **dendrograma** de tal manera que, dependiendo de solicitud del cliente, se puedan obtener distintos resultados.

Hallando la matriz de distancia

Recordemos que la entrada de los métodos jerárquicos es la matriz de distancia o disimilaridad. En R es sencillo calcular esta distancia usando el método `dist`.

```
# Copiamos el dataset en una variable nueva
entrada.num = datos

# Eliminamos la columna clase para obtener la matriz de distancia adecuada
entrada.num$Class = NULL

# De DataFrame a Matrix
entrada.num = as.matrix(entrada.num)

# Matriz de distancia, ?dist para otras opciones distinta a norma 2
distancia = dist(entrada.num)
```

Llamando al método Hclust

Una vez calculada la matriz de distancia podemos hacer el clustering llamando al método `hclust`.

```
# Método por defecto es complete link
metodo = "complete"
cluster = hclust(distancia, method = metodo)
```

El Dendrograma

Recordemos que esta estructura tiene en su eje X todas y cada una de las instancias del dataset y en el eje Y la medida de disimilaridad.

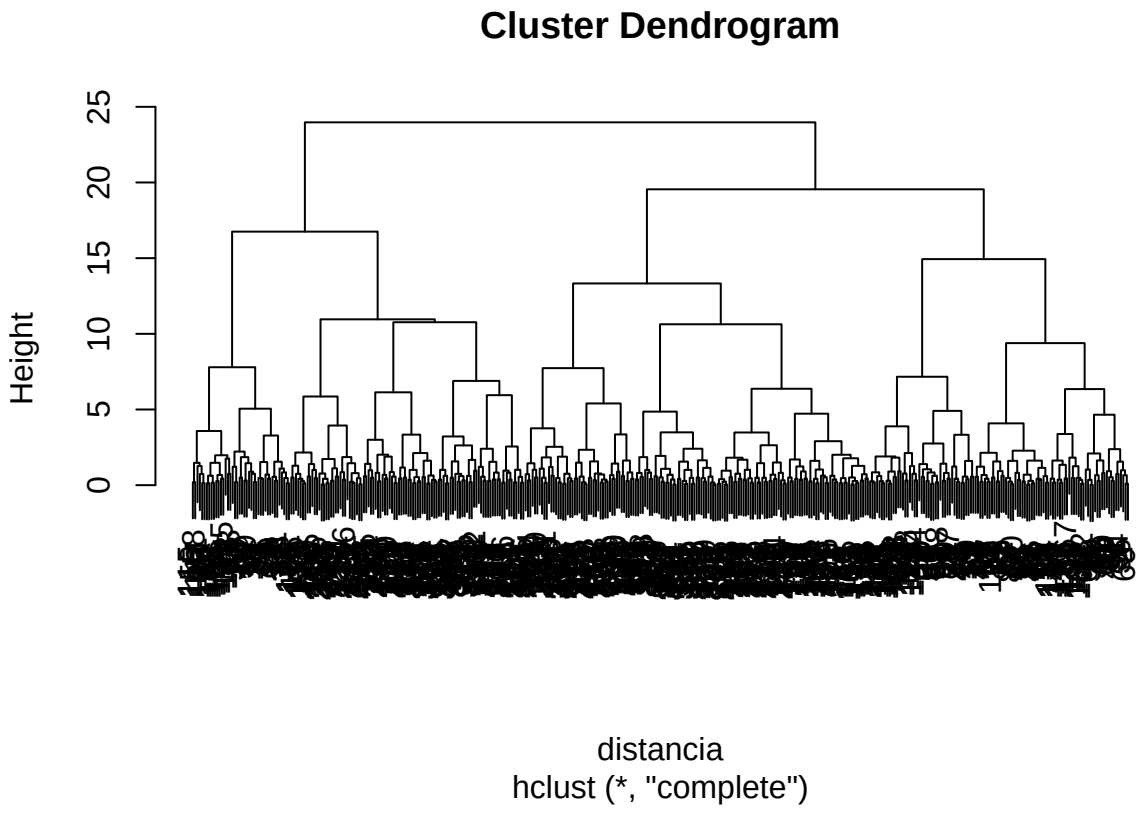
Dicho esto, a medida que el eje Y aumenta, conseguimos menos intersecciones en el árbol generado. Nuestra tarea puede resumirse, en consecuencia, a dos escenarios:

1. Dada una altura h (una medida de disimilaridad) determinar el número de clústers que se obtienen.
2. Dado un número de clústers k determinar la altura requerida para que tengamos el número de clúster k.

Graficando

Para graficar el dendrograma simplemente llamamos a la función plot de R:

```
plot(cluster)
```



Usando el dendrograma

Existen dos maneras de usar el dendrograma en R.

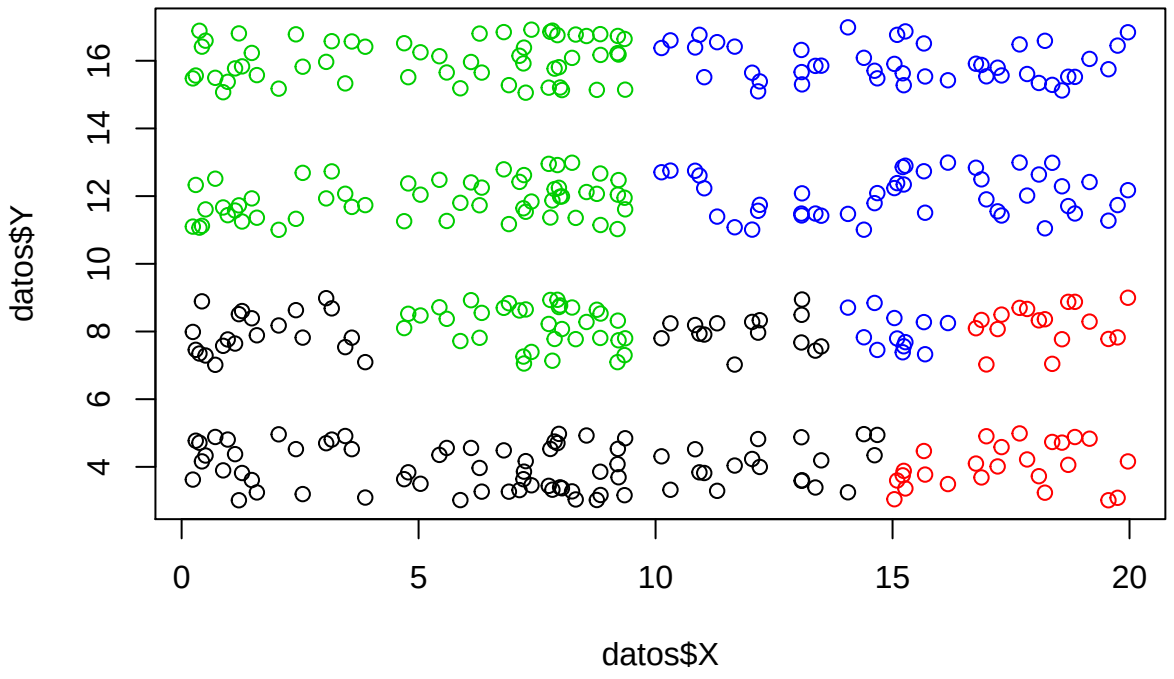
1. Usamos el método `cutree` sin necesidad de usar una estructura auxiliar.
2. Generamos una estructura auxiliar del tipo **dendrogram** para mejor manipulación.

```
# Seleccionamos el número de clases que queremos y cortamos el árbol en esa altural.
nclases = 4

# Cortamos
corte = cutree(cluster, k=nclases)

# Qué hay en corte?
head(corte)
```

```
## 1 2 3 4 5 6
## 1 1 1 1 1 2
```

Como vemos, usando el primer método obtenemos un vector en dónde cada instancia fue seleccionada a cada clúster. Así, graficamos con los colores referentes a los clústers para analizar resultados.

```
plot(x = datos$X,
     y = datos$Y,
     col = corte)
```



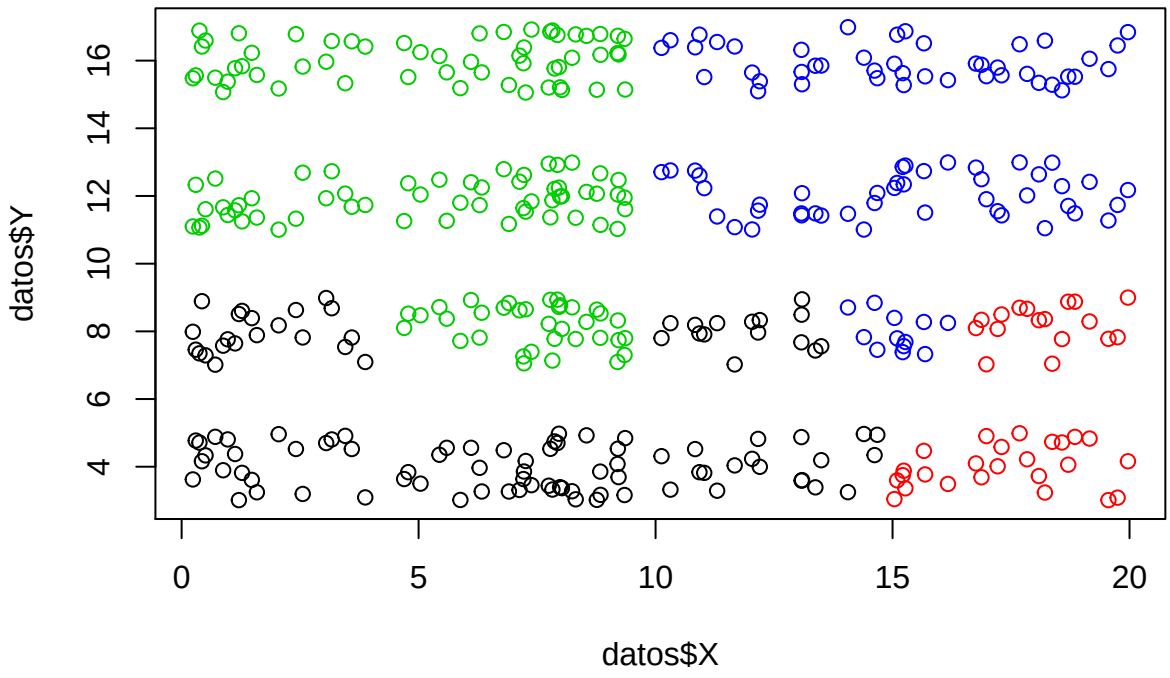
Si quisiéramos, en contraparte, los clústers asociados a una altura, simplemente usamos el parámetro h en vez de k en el método `cutree`.

Si vemos el gráfico del dendrograma, al cortar en la altura 15 deberían generarse solamente 4 colores (4 clústers).

```
# Cortamos por altura ahora
corte = cutree(cluster, h=15)

# Verificamos cuántos clúster tenemos
unique(corte)
```

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

```
# Graficamos nuevamente
 plot(x = datos$X, y = datos$Y, col = corte)
```



Es de **suma** importancia recalcar el hecho de que toda instancia debe estar en el mismo clúster asignado sin importar si cortamos por altura o por número de clúster. De hecho, se puede apreciar que ambas gráficas son idénticas.

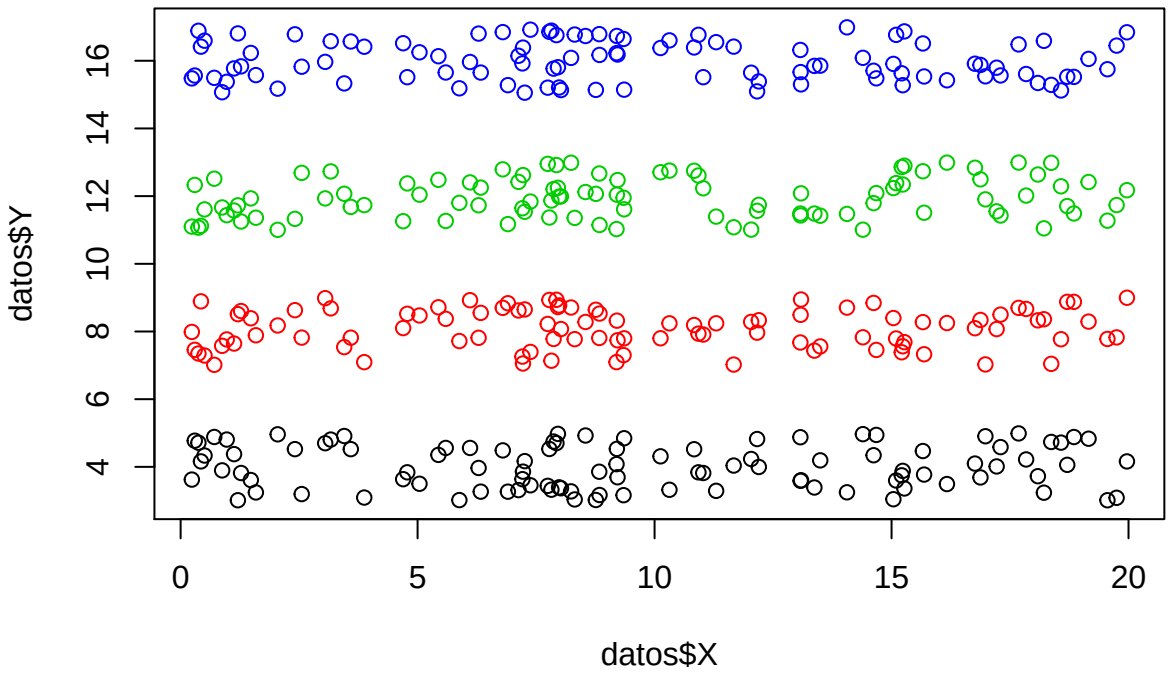
Ahora, es notable que el método parece no estar funcionando. Grafiquemos la matriz de confusión:

```
# Comparamos datos$class con la salida del método
table(datos$class, corte)
```

```
##    corte
##      1  2  3  4
##   1 75 25  0  0
##   2 36 17 35 12
##   3  0  0 56 44
##   4  0  0 56 44
```

Esto es debido a que estamos escogiendo (como vimos en clase) la distancia *complete link*. Dada la estructura del clúster, debería funcionar mejor *single link*. Verificamos:

```
# Cambiamos método a single link
metodo = "single"
cluster = hclust(distancia, method = metodo)
corte = cutree(cluster, k=4)

# Graficamos con single link
 plot(x = datos$X, y = datos$Y, col = corte)
```



Parece estar funcionando, cómo verificamos? Nuevamente, calculamos la matriz de confusión:

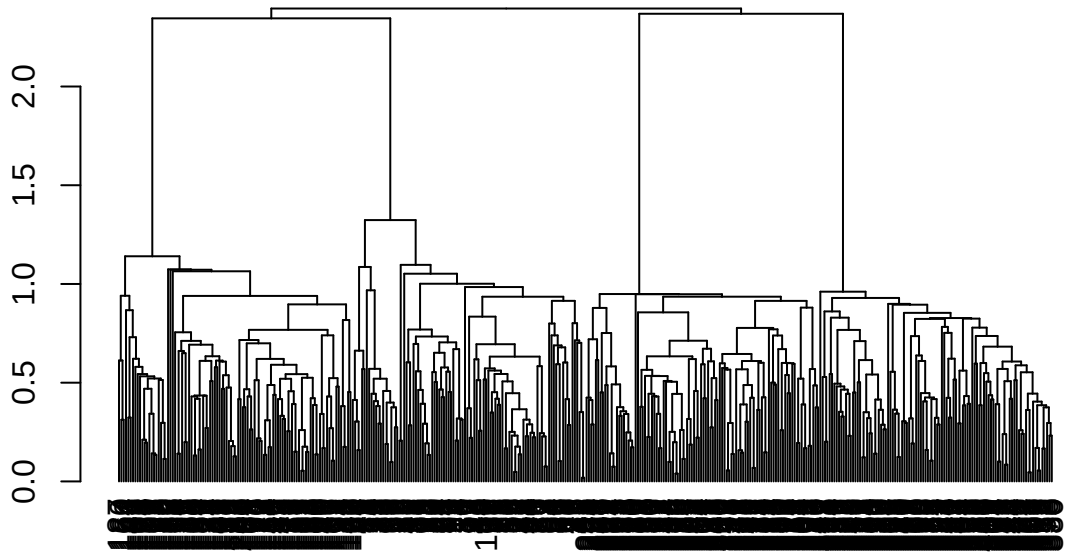
```
  table(datos$class, corte)
```

```
##    corte
##       1   2   3   4
##   1 100   0   0   0
##   2   0 100   0   0
##   3   0   0 100   0
##   4   0   0   0 100
```

Cada elemento de cada clase pertence al mismo clúster.

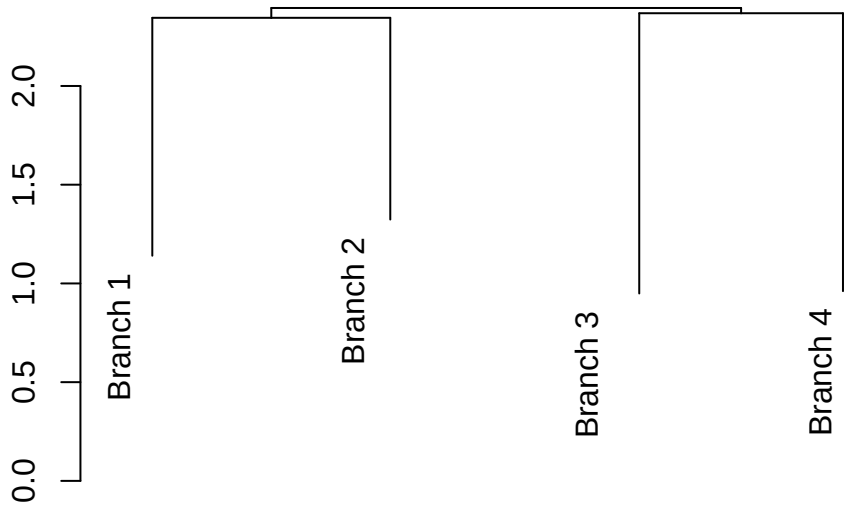
Segundo método: A diferencia del método anterior, simplemente usamos una tipo de dato auxiliar llamado **dendrogram**. En este, podemos obtener el árbol asociado luego de realizar el corte:

```
# Verifiquemos el dendrograma
cluster = hclust(distancia, method = metodo)
dendrogram = as.dendrogram(cluster)
plot(dendrogram)
```



Ahora, cortamos por altura y reutilizamos la estructura dendrogram.

```
cortes = cut(dendrogram, h = 1.5)$upper
plot(cortes)
```



Para reflexionar Qué ocurre si seleccionamos el atributo lower en vez de upper?

Referencias

Material diseñado usando las librerías de referencia de `kmeans` y `hclust` inherentes a la versión de R.

Los datasets son de propiedad propia.