Minería de datos: PEC2 - Métodos no supervisados

Autor: Juan Luis Acebal Rico

Noviembre 2024

Contents

Ejemplo guiado 1.1

Método de agregación k-means con datos autogenerados

En este ejemplo vamos a generar un conjunto de muestras aleatorias para posteriormente usar el algoritmo *k-means* para agruparlas. Se crearán las muestras alrededor de dos puntos concretos. Por lo tanto, lo lógico será agrupar en dos clústers.

Puesto que inicialmente, en un problema real, no se conoce cual es el número más idóneo de clústers k, vamos a probar primero con dos (el valor óptimo) y posteriormente con 4 y 8 clústers. Para evaluar la calidad de cada proceso de agrupación vamos a usar la silueta media. La silueta de cada muestra evalúa como de bien o mal está clasificada la muestra en el clúster al que ha sido asignada. Para ello se usa una fórmula que tiene en cuenta la distancia a las muestras de su clúster y la distancia a las muestras del clúster vecino más cercano.

A la hora de probar el código que se muestra, es importante tener en cuenta que las muestras se generan de forma aleatoria y también que el algoritmo *k-means* tiene una inicialización aleatoria. Por lo tanto, en cada ejecución se obtendrá unos resultados ligeramente diferentes.

Lo primero que hacemos es cargar la librería cluster que contiene las funciones que se necesitan

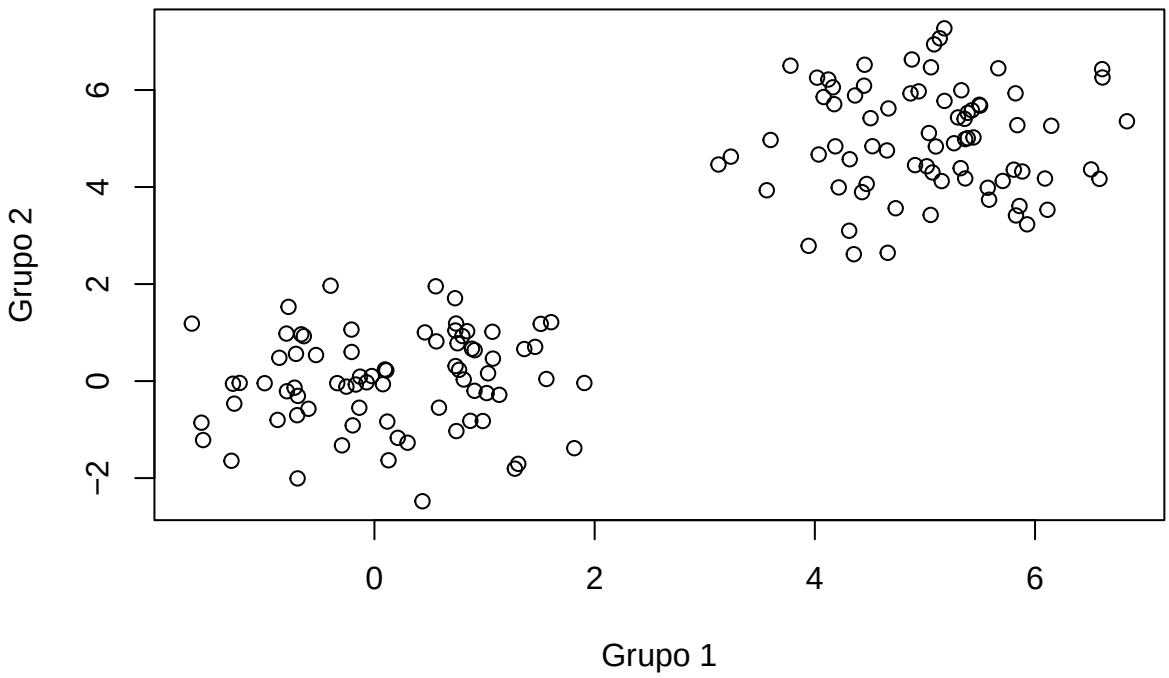
```
if (!require('cluster')) install.packages('cluster')
library(cluster)
```

Generamos las muestras de forma aleatoria tomando como centro los puntos [0,0] y [5,5].

```
n <- 150 # número de muestras
p <- 2   # dimensión

sigma <- 1 # varianza de la distribución
mean1 <- 0 # centro del primer grupo
mean2 <- 5 # centro del segundo grupo

n1 <- round(n/2) # número de muestras del primer grupo
n2 <- round(n/2) # número de muestras del segundo grupo

x1 <- matrix(rnorm(n1*p,mean=mean1,sd=sigma),n1,p)
x2 <- matrix(rnorm(n2*p,mean=mean2,sd=sigma),n2,p)
```

Juntamos todas las muestras generadas y las mostramos en una gráfica

```
x  <- rbind(x1,x2)
plot (x, xlab="Grupo 1", ylab="Grupo 2")
```



Como se puede comprobar las muestras están claramente separadas en dos grupos. Si se quiere complicar el problema se puede modificar los puntos centrales (mean1 y mean2) haciendo que estén más próximos y/o ampliar la varianza (sigma) para que las muestras estén más dispersas.

A continuación vamos a aplicar el algoritmo *k-means* con 2, 4 y 8 clústers

```
# Función para ajustar k-means y mostrar los resultados
ajustar_kmeans <- function(k, vdata) {
  fit <- kmeans(vdata, centers = k)
  return(fit$cluster)
}

# Ajustos amb k = 2, 4, 8
clusters_2 <- ajustar_kmeans(2, x)
clusters_4 <- ajustar_kmeans(4, x)
clusters_8 <- ajustar_kmeans(8, x)
```

Las variables y_cluster2, y_cluster4 e y_cluster8 contienen para cada muestra el identificador del clúster a las que han sido asignadas. Por ejemplo, en el caso de los k=2 las muestras se han asignado al clúster 1 ó al 2

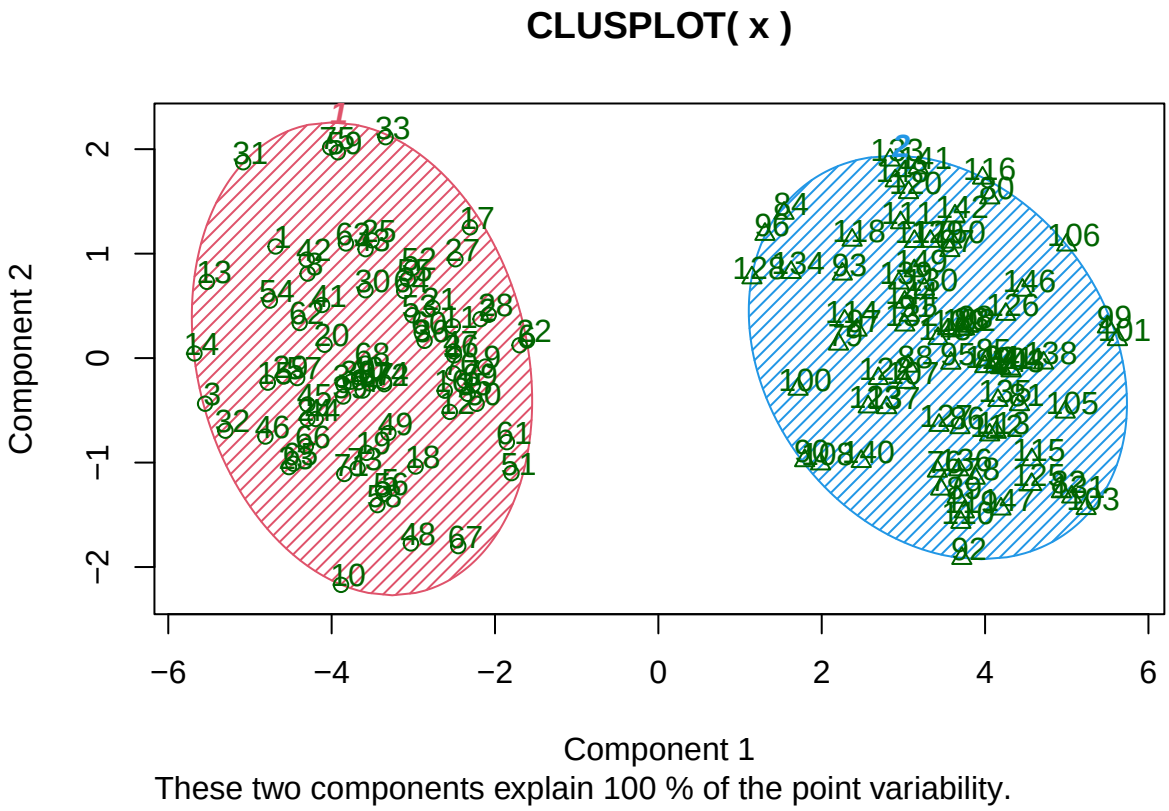
```
clusters_2
```

```
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [75] 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

```
## [112] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [149] 2 2
```

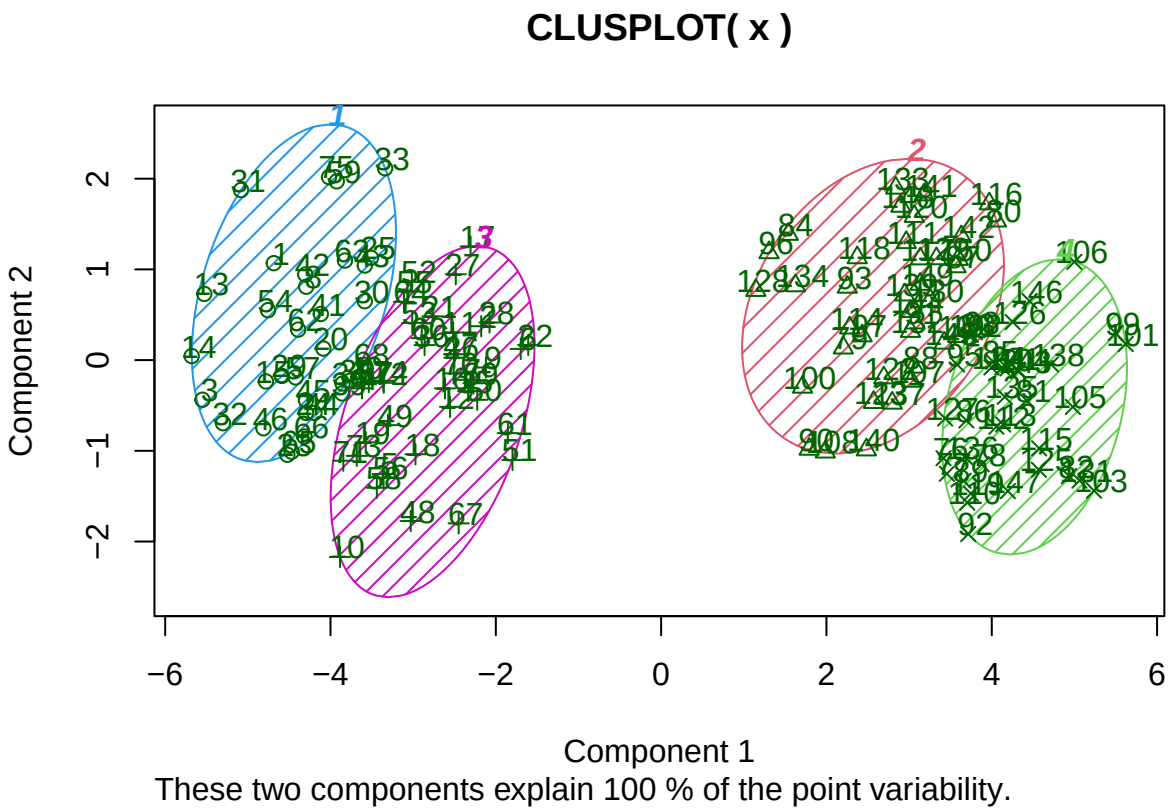
Para visualizar los clústers podemos usar la función clusplot. Vemos la agrupación con 2 clústers y observemos como prácticamente no hay valores extremos y realmente los dos clústers generados son homogéneos.

```
clusplot(x, clusters_2, color=TRUE, shade=TRUE, labels=2, lines=0)
```



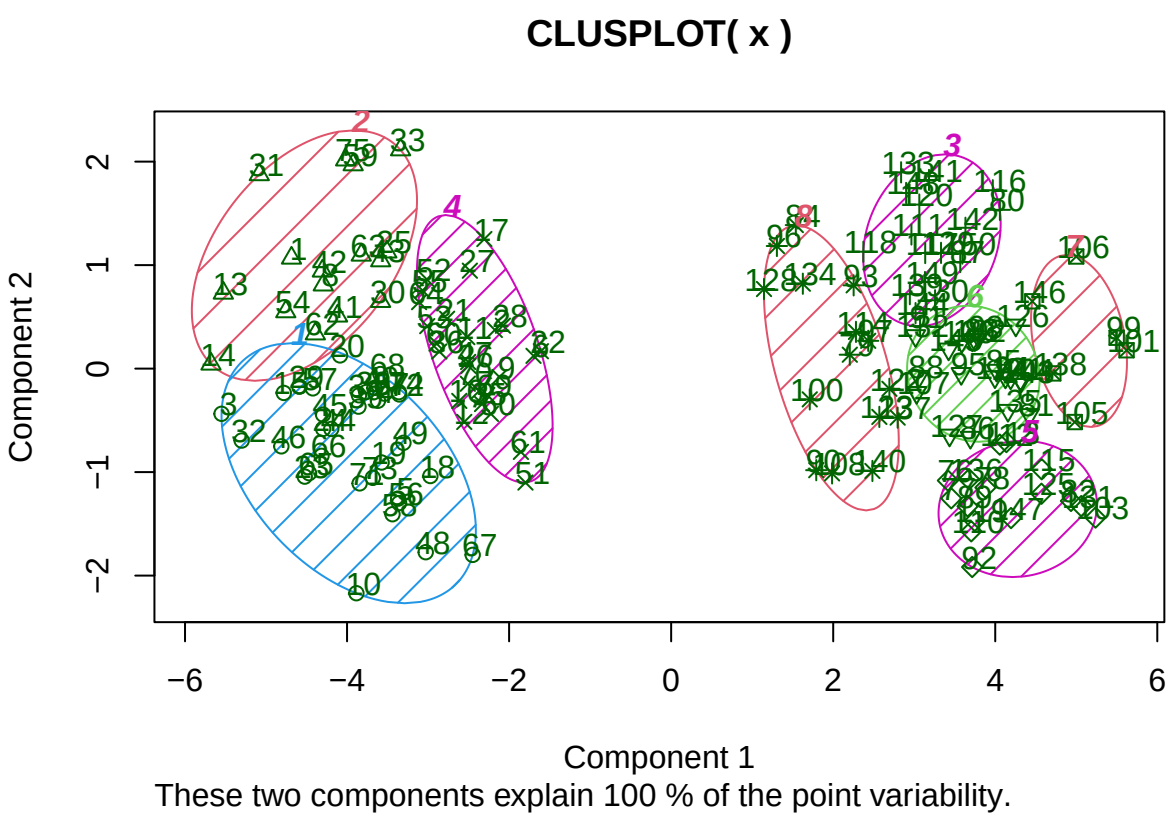
con 4 observamos como el clúster de la izquierda lo ha dividido en 3.

```
clusplot(x, clusters_4, color=TRUE, shade=TRUE, labels=2, lines=0)
```



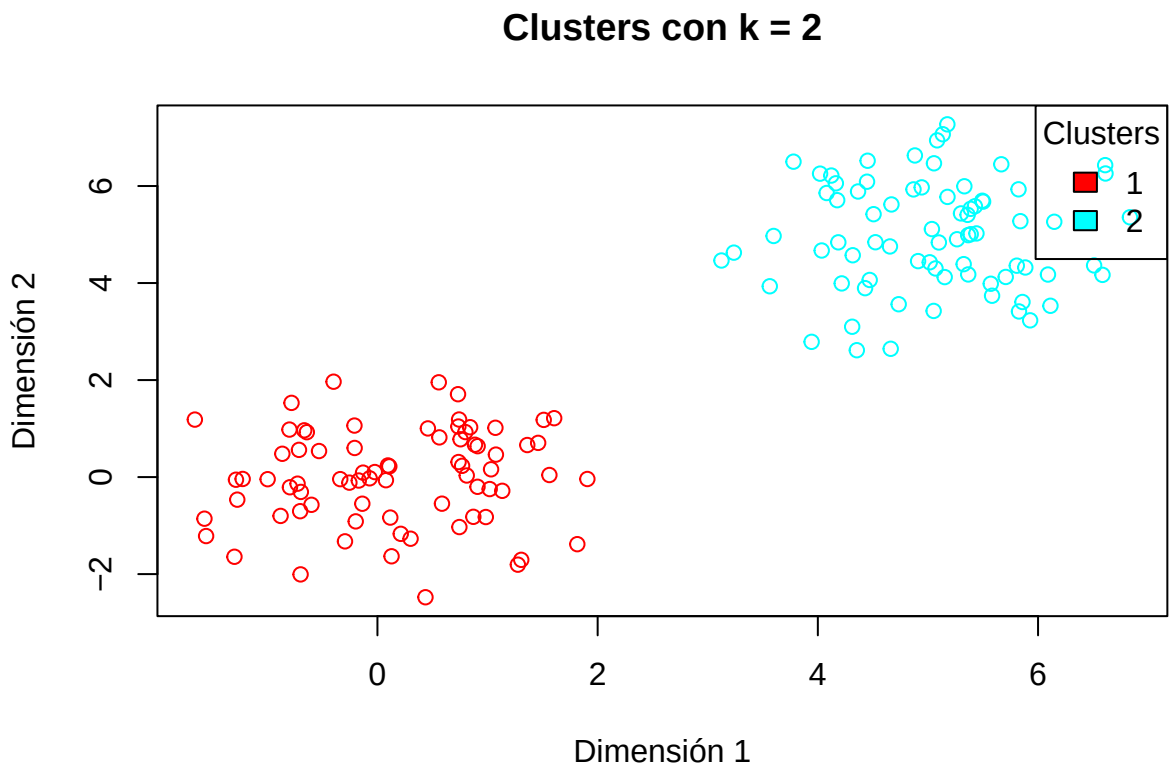
y con 8. El algoritmo obedece y nos genera 8 clústers aunque como se aprecia visualmente no tenga demasiada consistencia.

```
clusplot(x, clusters_8, color=TRUE, shade=TRUE, labels=2, lines=0)
```



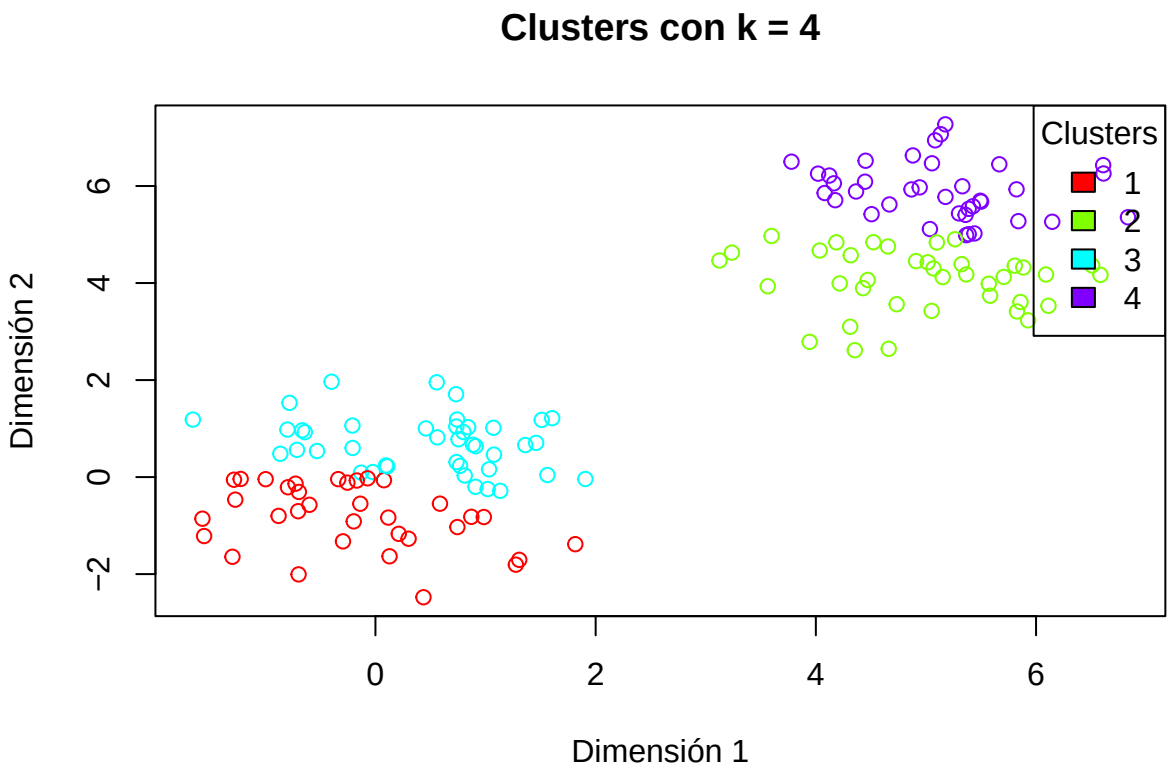
También podemos visualizar el resultado del proceso de agrupamiento con el siguiente código para el caso de 2 clústers. El uso de colores facilita la identificación visual de clústers.

```
# Función para graficar los resultados de los clusters
graficar_clusters <- function(vdata, y_cluster, k) {
  colors <- rainbow(k)
  plot(vdata, col = colors[y_cluster], xlab = "Dimensión 1", ylab = "Dimensión 2",
       main = paste("Clusters con k =", k))
  legend("topright", legend = 1:k, fill = colors, title = "Clusters")
}

# Gráfico para k = 2
graficar_clusters(x, clusters_2, 2)
```



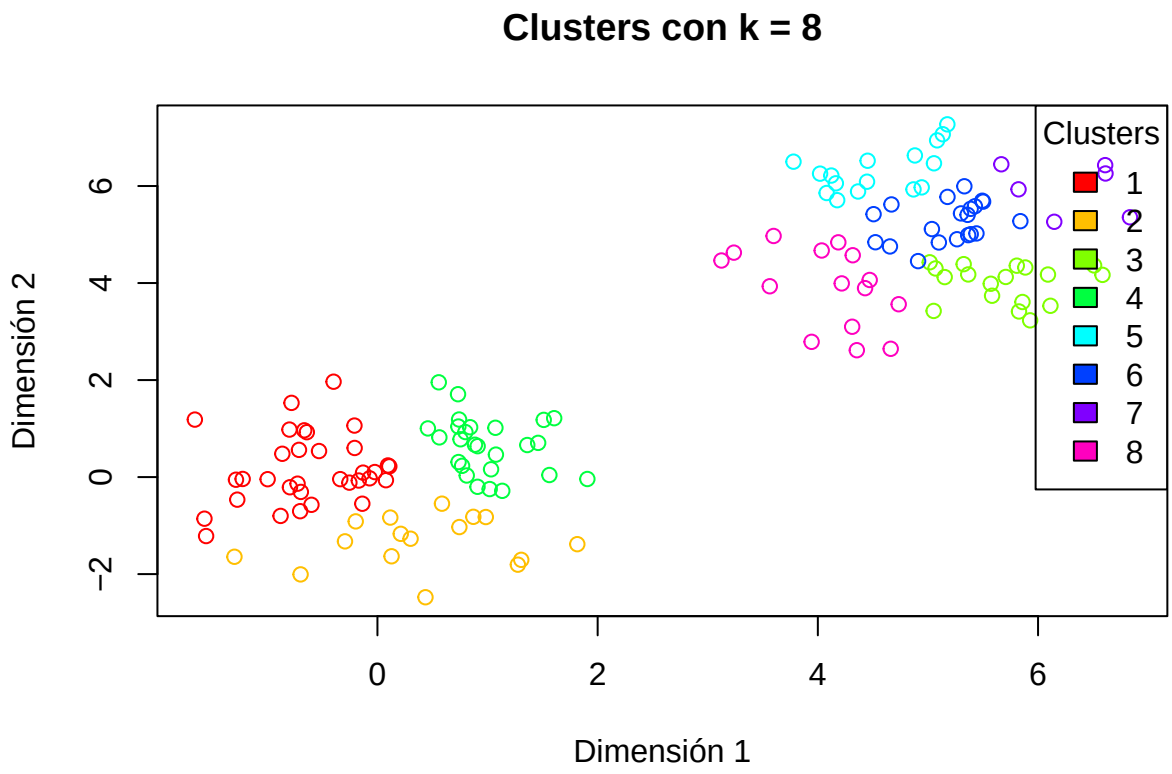
para 4

```
# Gráfico para k = 4
graficar_clusters(x, clusters_4, 4)
```



y para 8

```
# Gráfico para k = 8
graficar_clusters(x, clusters_8, 8)
```



Ahora vamos a evaluar la calidad del proceso de agregación. Para ello usaremos la función silhouette que calcula la silueta de cada muestra

```
# Función para calcular y mostrar la silueta
calcular_silueta <- function(y_cluster, vdata) {
  distances <- daisy(vdata)
  silueta <- silhouette(y_cluster, distances)
  mean_sil <- mean(silueta[, 3])
  return(mean_sil)
}

sil_2 <- calcular_silueta(clusters_2, x)
sil_4 <- calcular_silueta(clusters_4, x)
sil_8 <- calcular_silueta(clusters_8, x)
```

La función silhouette devuelve para cada muestra, el clúster dónde ha sido asignado, el clúster vecino y el valor de la silueta. Por lo tanto, calculando la media de la tercera columna podemos obtener una estimación de la calidad del agrupamiento

```
# Mostrar valores de siluetas medianas
cat("Silueta mediana para k=2:", sil_2, "\n")
```

```
## Silueta mediana para k=2: 0.7538244
```

```
cat("Silueta mediana para k=4:", sil_4, "\n")
```

```
## Silueta mediana para k=4: 0.3672782
```

```
cat("Silueta mediana para k=8:", sil_8, "\n")
```

```
## Silueta mediana para k=8: 0.3813253
```

Como se puede comprobar, agrupar con dos clúster es mejor que en 4 o en 8, lo cual es lógico teniendo en cuenta como se han generado los datos.

Una buena práctica para entender mejor el juego de datos, consiste en poner nombre a cada uno de los clústers identificados. Lo veremos más claramente en el siguiente ejemplo que parte de datos reales.

Ejemplo guiado 1.2

Método de agregación k-means con datos reales

A continuación vamos a ver otro ejemplo de cómo se usan los modelos de agregación. Para ello usaremos el data set **penguins** contenido en el paquete R **palmerpenguins**. Esta base de datos se encuentra descrita en https://cran.r-project.org/web/packages/palmerpenguins/index.html y contiene mediciones de tamaño, observaciones de puestas y proporciones de isótopos sanguíneos de tres especies de pingüinos observadas en tres islas del archipiélago Palmer, en la Antártida, durante un período de estudio de tres años.

Este dataset está previamente trabajado para que los datos estén limpios y sin errores. De no ser así antes de nada deberíamos buscar errores, valores nulos u *outliers*. Deberíamos tratar de discretizar o eliminar columnas. Incluso realizar este último paso varias veces para comprobar los diferentes resultados y elegir el que mejor rendimiento nos dé. De todos modos contiene algún valor nulo que procederemos a ignorar.

Vamos a visualizar la estructura y resumen de los datos

```
if (!require('palmerpenguins')) install.packages('palmerpenguins')
library(palmerpenguins)
palmerpenguins::penguins
```

```
## # A tibble: 344 x 8
##    species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
##    <fct>   <fct>              <dbl>         <dbl>             <int>       <int>
##  1 Adelie  Torgersen           39.1          18.7               181        3750
##  2 Adelie  Torgersen           39.5          17.4               186        3800
##  3 Adelie  Torgersen           40.3          18                 195        3250
##  4 Adelie  Torgersen           NA            NA                  NA          NA
##  5 Adelie  Torgersen           36.7          19.3               193        3450
##  6 Adelie  Torgersen           39.3          20.6               190        3650
##  7 Adelie  Torgersen           38.9          17.8               181        3625
##  8 Adelie  Torgersen           39.2          19.6               195        4675
##  9 Adelie  Torgersen           34.1          18.1               193        3475
```

```
## 10 Adelie  Torgersen           42            20.2               190        4250
## # i 334 more rows
## # i 2 more variables: sex <fct>, year <int>
```

```
summary(penguins)
```

```
##       species          island    bill_length_mm  bill_depth_mm
##  Adelie   :152   Biscoe   :168   Min.   :32.10   Min.   :13.10
##  Chinstrap: 68   Dream    :124   1st Qu.:39.23   1st Qu.:15.60
##  Gentoo   :124   Torgersen: 52   Median :44.45   Median :17.30
##                                  Mean   :43.92   Mean   :17.15
##                                  3rd Qu.:48.50   3rd Qu.:18.70
##                                  Max.   :59.60   Max.   :21.50
##                                  NA's   :2       NA's   :2
##  flipper_length_mm  body_mass_g       sex           year
##  Min.   :172.0     Min.   :2700   female:165   Min.   :2007
##  1st Qu.:190.0     1st Qu.:3550   male  :168   1st Qu.:2007
##  Median :197.0     Median :4050   NA's  : 11   Median :2008
##  Mean   :200.9     Mean   :4202                Mean   :2008
##  3rd Qu.:213.0     3rd Qu.:4750                3rd Qu.:2009
##  Max.   :231.0     Max.   :6300                Max.   :2009
##  NA's   :2         NA's   :2
```

Como se puede comprobar, esta base de datos está pensada para problemas de clasificación supervisada que pretende clasificar cada tipo de pingüino en una de las tres clases o especies existentes (Adelie, Gentoo o Chinstrap). Como en este ejemplo vamos a usar un método no supervisado, transformaremos el problema supervisado original en uno **no supervisado**. Para conseguirlo no usaremos la columna *species*, que es la variable que se quiere predecir. Por lo tanto, intentaremos encontrar agrupaciones usando únicamente los cuatro atributos numéricos que caracterizan a cada especie de pingüino.
x <- na.omit(penguins[,3:6]) Cargamos los datos y nos quedamos únicamente con las cuatro columnas que definen a cada especie.

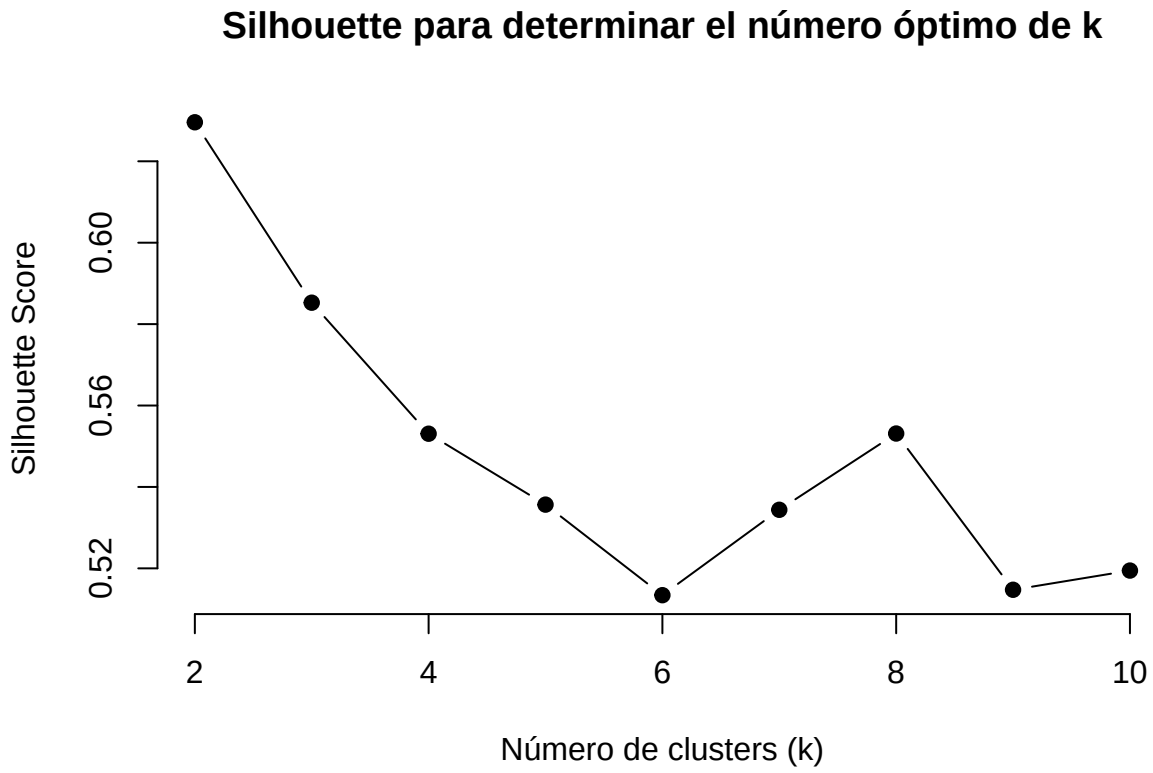
```
x_clean <- na.omit(penguins[,c(1,3,4,5,6)])
x_species <- x_clean[,1]
x <- x_clean[,2:5]
```

Planteamos ahora un ejemplo más realista en el que inicialmente no conocemos el número óptimo de clústers. Empecemos probamos con varios valores.

```
# Silhouette method
max_k <- 10
silhouettes <- numeric(max_k)
for (k in 2:max_k) {
  y_cluster <- kmeans(x, centers = k)$cluster
  silhouettes[k] <- calcular_silueta(y_cluster, x)
}
```

Mostramos en un gráfica los valores de las siluetas media de cada prueba para comprobar que número de clústers es el mejor.

```
plot(2:max_k, silhouettes[2:max_k], type = "b", pch = 19, frame = FALSE,
     xlab = "Número de clusters (k)", ylab = "Silhouette Score",
     main = "Silhouette para determinar el número óptimo de k")
```



Los valores de la silueta pueden fluctuar en el rango [-1,1], siendo valores cercanos a 1 indicativos de homogeneidad en los grupos y por el contrario valores de la silueta cercanos a -1 son indicativos de poca homogeneidad en los grupos, de modo que quisiéramos encontrarnos en un rango razonablemente cerca de 1.

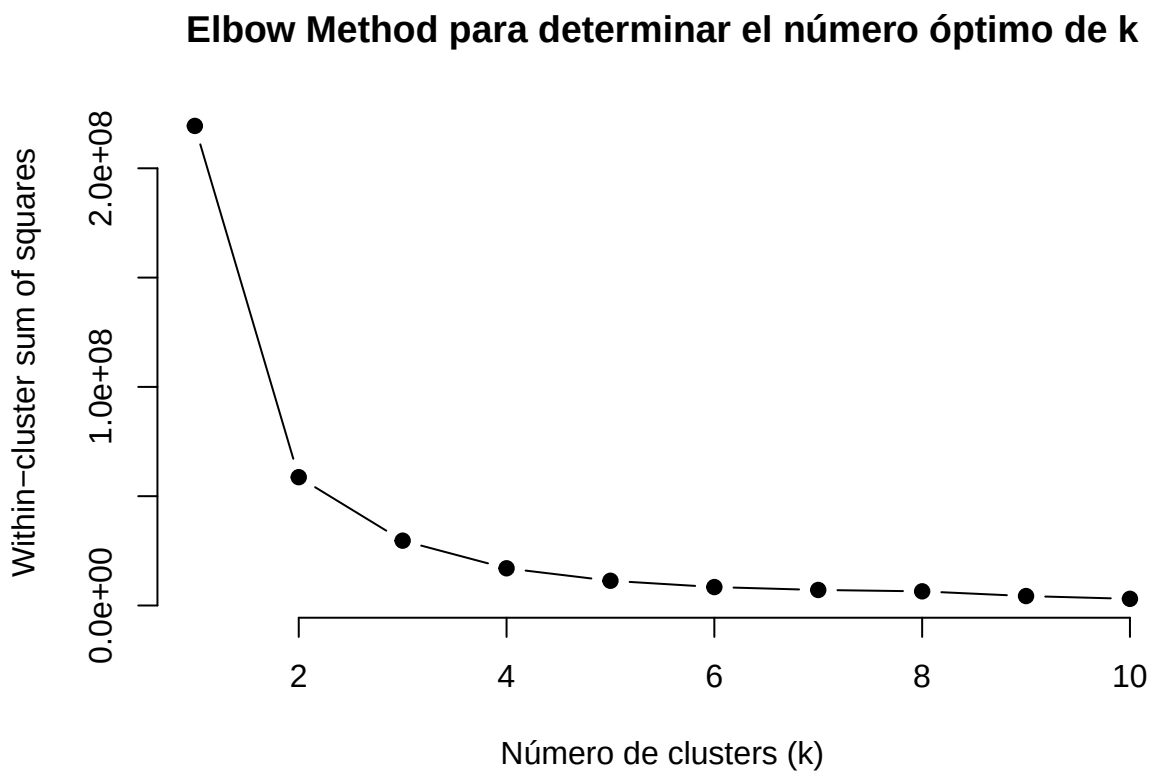
En el caso de nuestro juego de datos, a pesar de que uno esperaría obtener un valor óptimo para k = 3, parece que del gráfico se desprende que es mejor k = 2.

Sin embargo, merece la pena observar que a partir de k = 3 la pérdida de homogeneidad es relativamente pequeña ya que se mantiene estable en el rango [0.50, 0.56]. Este hecho podría ser un argumento para seleccionar k = 3.

Otra forma de evaluar cual es el mejor número de clústers es considerar el mejor modelo, aquel que ofrece la menor suma de los cuadrados de las distancias de los puntos de cada grupo con respecto a su centro (withinss), con la mayor separación entre centros de grupos (betweenss). Como se puede comprobar es una idea conceptualmente similar a la silueta. Una manera común de hacer la selección del número de clústers consiste en aplicar el método *elbow* (codo), que no es más que la selección del número de clústers en base a la inspección de la gráfica que se obtiene al iterar con el mismo conjunto de datos para distintos valores del número de clústers. Se seleccionará el valor que se encuentra en el "codo" de la curva.

```
# Función para calcular la inercia intracluster (Within-cluster sum of squares)
inercia_intracluster <- function(vdata, max_k) {
  wss <- numeric(max_k)
```

```
  for (k in 1:max_k) {
    wss[k] <- sum(kmeans(vdata, centers = k)$withinss)
  }
  return(wss)
}

# Elbow method
wss <- inercia_intracluster(x, max_k)
plot(1:max_k, wss, type = "b", pch = 19, frame = FALSE,
     xlab = "Número de clusters (k)", ylab = "Within-cluster sum of squares",
     main = "Elbow Method para determinar el número óptimo de k")
```



En este caso el número óptimo de clústers son 4 que es cuando la curva comienza a estabilizarse.

También se puede usar la función *kmeansruns* del paquete **fpc** que ejecuta el algoritmo kmeans con un conjunto de valores, para después seleccionar el valor del número de clústers que mejor funcione de acuerdo a dos criterios: la silueta media ("asw") y *Calinski-Harabasz* ("ch").

```
if (!require('fpc')) install.packages('fpc')
library(fpc)
fit_ch  <- kmeansruns(x, krange = 1:10, criterion = "ch")
fit_asw <- kmeansruns(x, krange = 1:10, criterion = "asw")
```

Podemos comprobar el valor con el que se ha obtenido el mejor resultado y también mostrar el resultado obtenido para todos los valores de k usando ambos criterios

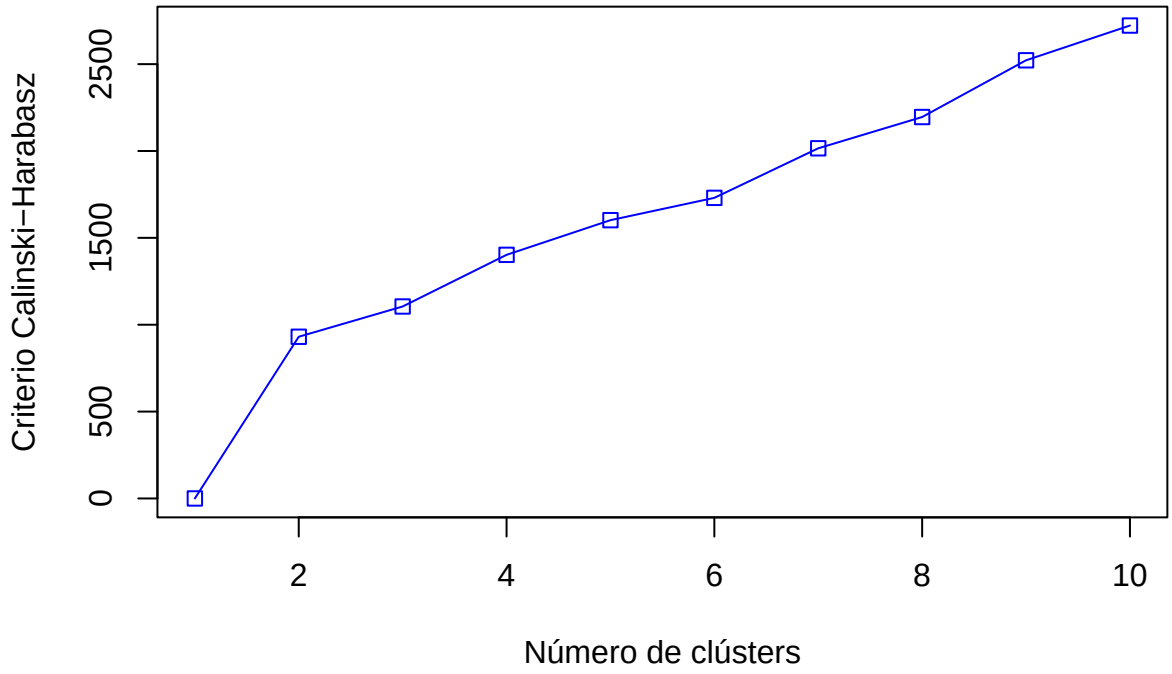
```
fit_ch$bestk
```

```
## [1] 10
```

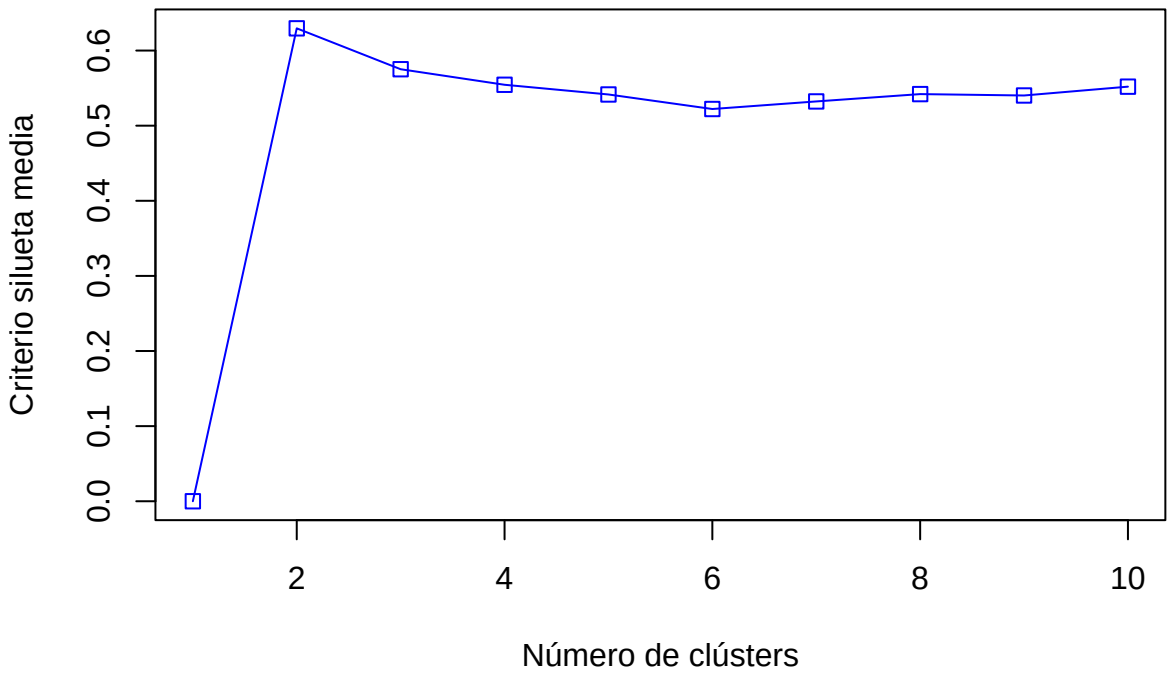
```
fit_asw$bestk
```

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

```
plot(1:10,fit_ch$crit,type="o",col="blue",pch=0,xlab="Número de clústers",ylab="Criterio Calinski-Haraba
```




```
plot(1:10,fit_asw$crit,type="o",col="blue",pch=0,xlab="Número de clústers",ylab="Criterio silueta media
```



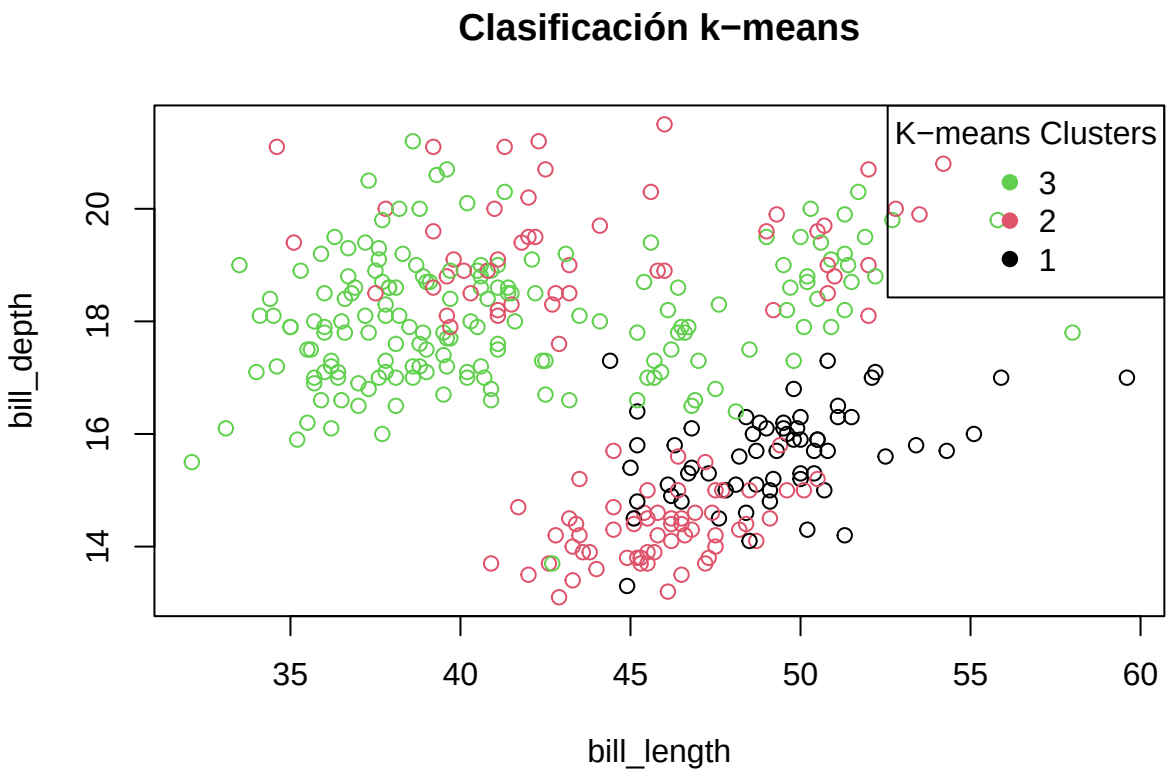
Los resultados son muy parecidos a los que hemos obtenido anteriormente. Con el criterio de la silueta media se obtienen dos clústers y con el *Calinski-Harabasz* se obtienen 3.

Como se ha comprobado, conocer el número óptimo de clústers no es un problema fácil. Tampoco lo es la evaluación de los modelos de agregación.

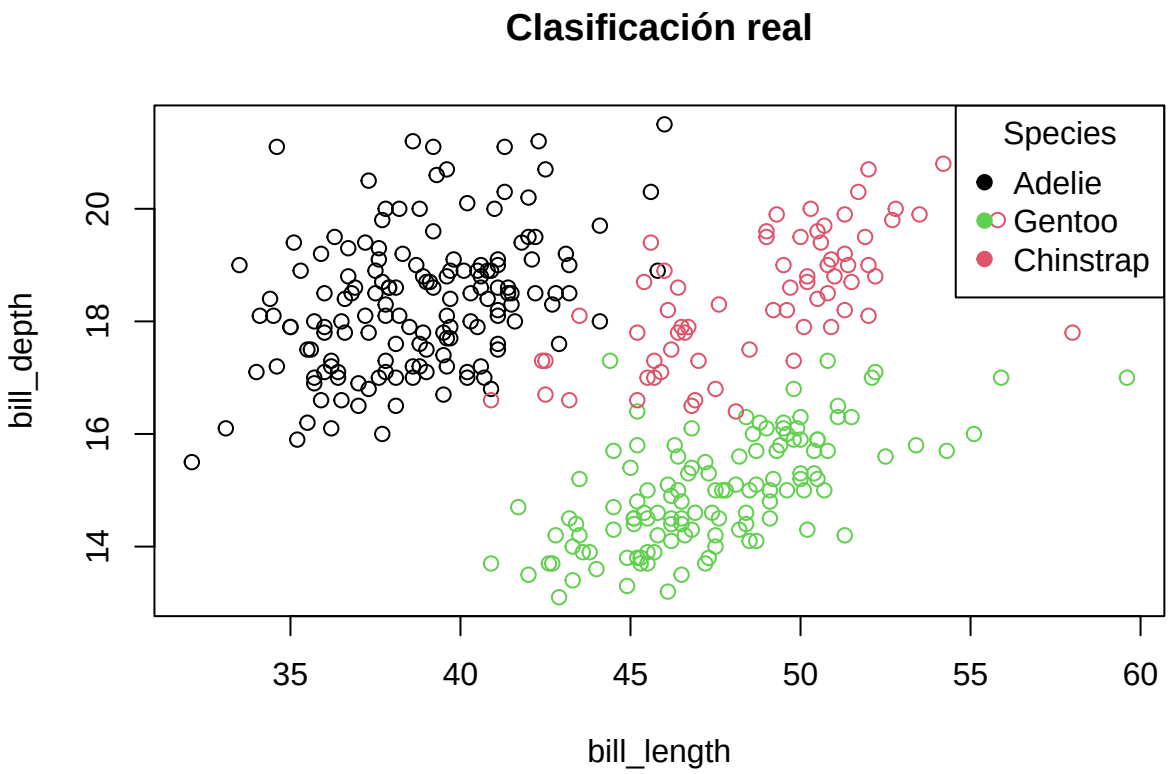
Como en el caso que estudiamos sabemos que los datos pueden ser agrupados en 3 clases o especies, vamos a ver cómo se ha comportado *kmeans* en el caso de pedirle 3 clústers. Para eso comparamos visualmente los campos dos a dos, con el valor real que sabemos está almacenado en el campo "species" del dataset original.

(Aclaramos que obviamente no acostumbra a pasar que conozcamos de forma previa el número de clústers óptimo. Este ejemplo lo planteamos con finalidades didácticas y con voluntad de experimentar)

```
penguins3clusters <- kmeans(x, 3)

# bill_length y bill_depth
plot(x[c(1,2)],
     col = penguins3clusters$cluster,    # Color by k-means cluster
     main = "Clasificación k-means",     # Plot title
     xlab = "bill_length",
     ylab = "bill_depth")

# Add a legend for the clusters
legend("topright",
       legend = unique(penguins3clusters$cluster), # Cluster labels
       col = unique(penguins3clusters$cluster),    # Colors corresponding to clusters
       pch = 19,                                   # Point symbol in the legend
       title = "K-means Clusters")
```

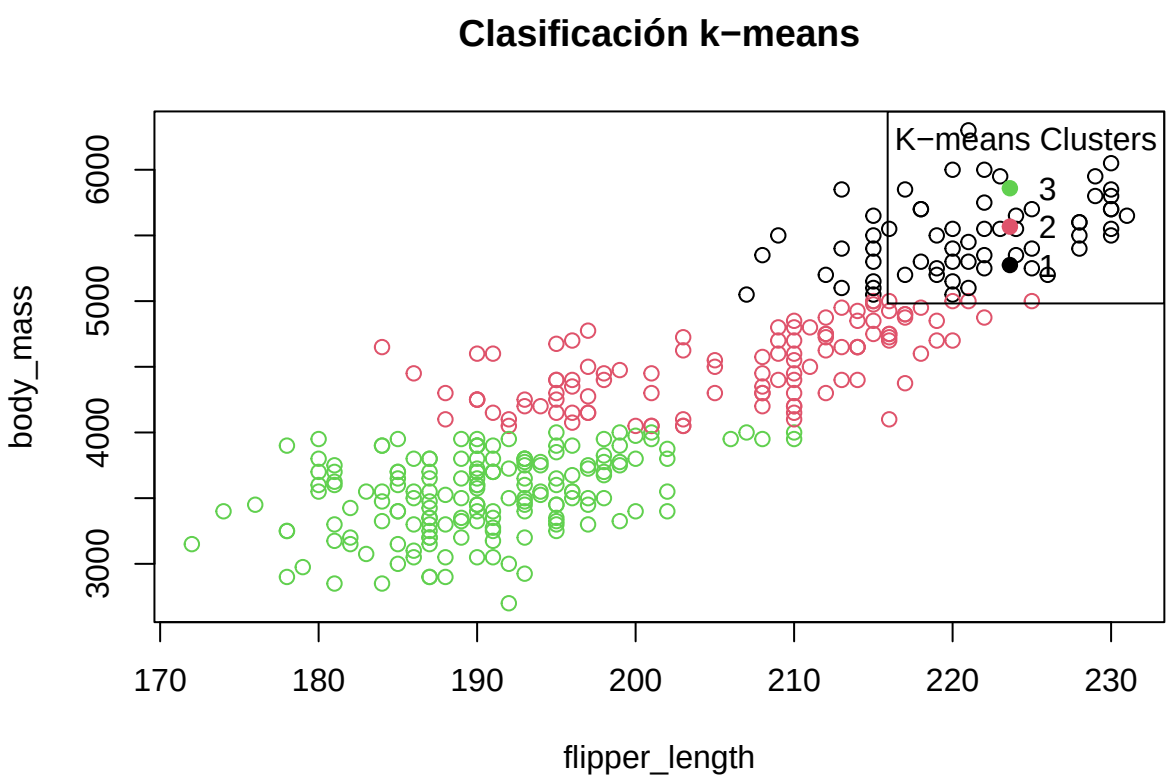



```r
# Plot with real species classification
plot(x[c(1,2)],
     col = as.factor(x_species$species), # Color by real species
     main = "Clasificación real",        # Plot title
     xlab = "bill_length",
     ylab = "bill_depth")

# Add a legend for species
legend("topright",
       legend = unique(x_species$species),        # Species labels
       col = unique(as.factor(x_species$species)), # Colors corresponding to species
       pch = 19,                                  # Point symbol in the legend
       title = "Species")
```

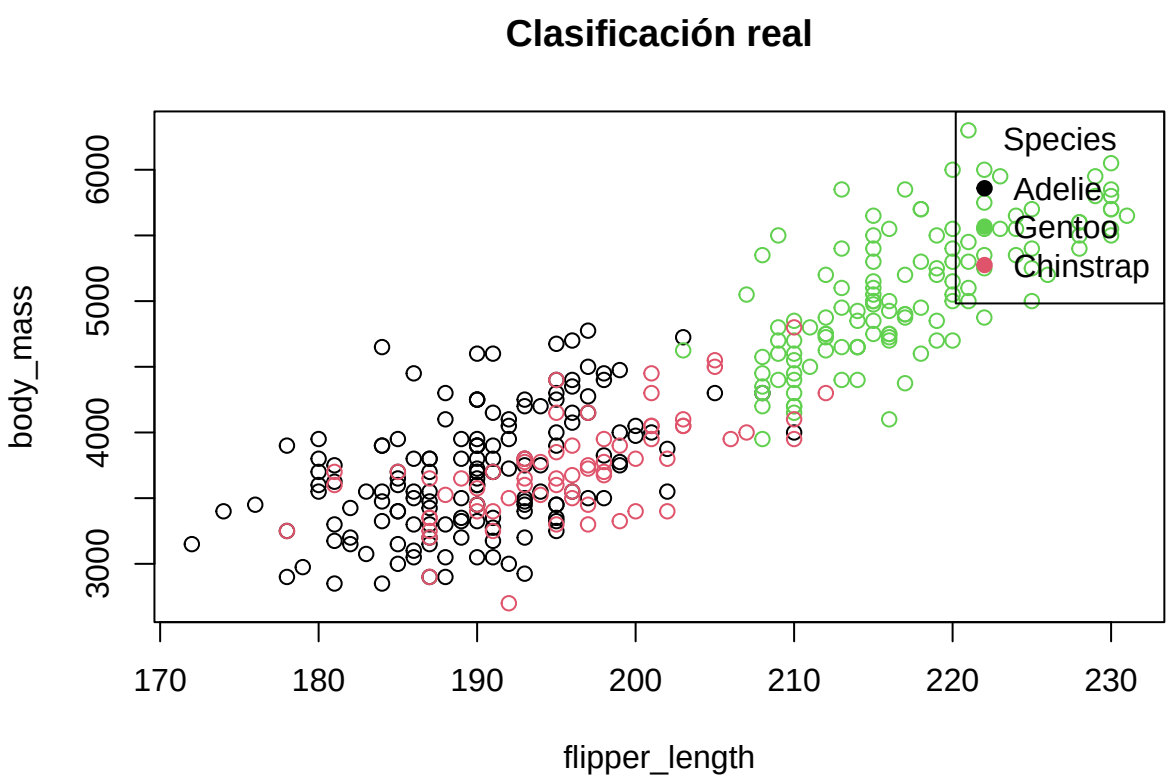


Podemos observar que *flipper_length* y *body_mass* no son buenos indicadores para diferenciar a las tres subespecies, dado que dos de las subespecies están demasiado mezcladas para poder diferenciar nada.

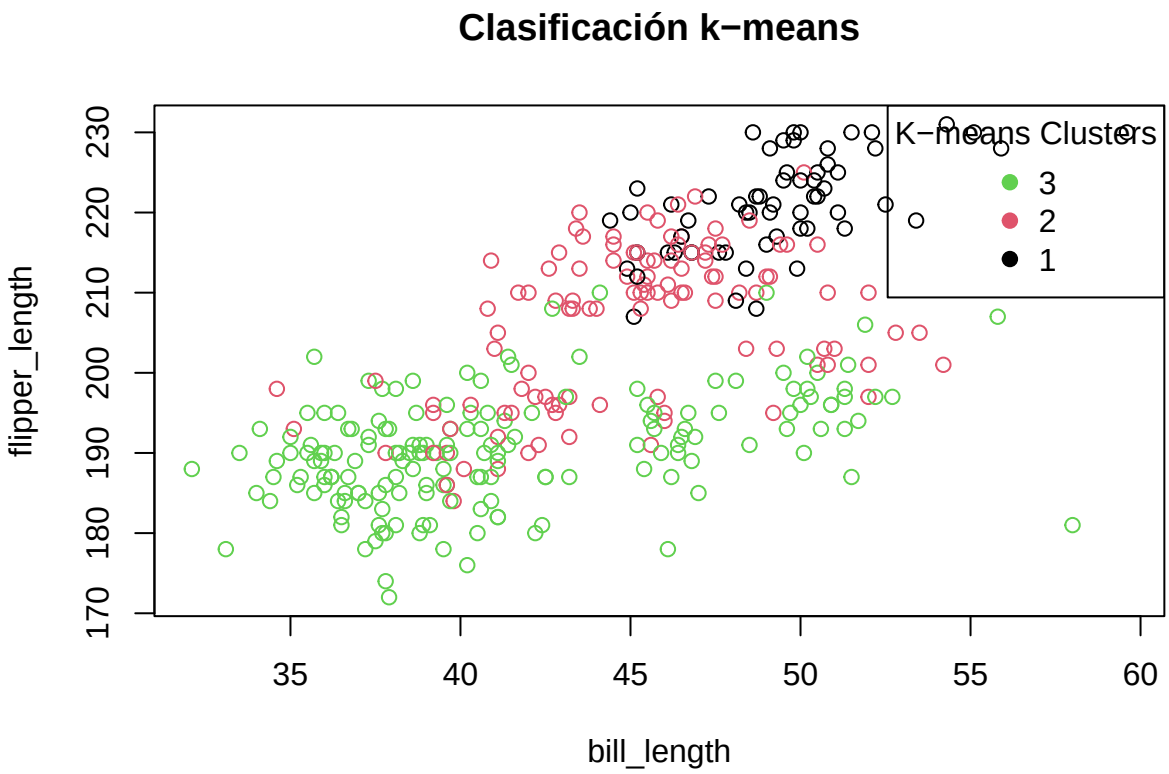
```
# flipper_length y body_mass
plot(x[c(3,4)],
     col = penguins3clusters$cluster,    # Color by k-means cluster
     main = "Clasificación k-means",     # Plot title
     xlab = "flipper_length",
     ylab = "body_mass")

# Add a legend for the clusters
legend("topright",
       legend = unique(penguins3clusters$cluster), # Cluster labels
       col = unique(penguins3clusters$cluster),    # Colors corresponding to clusters
       pch = 19,                                   # Point symbol in the legend
       title = "K-means Clusters")
```



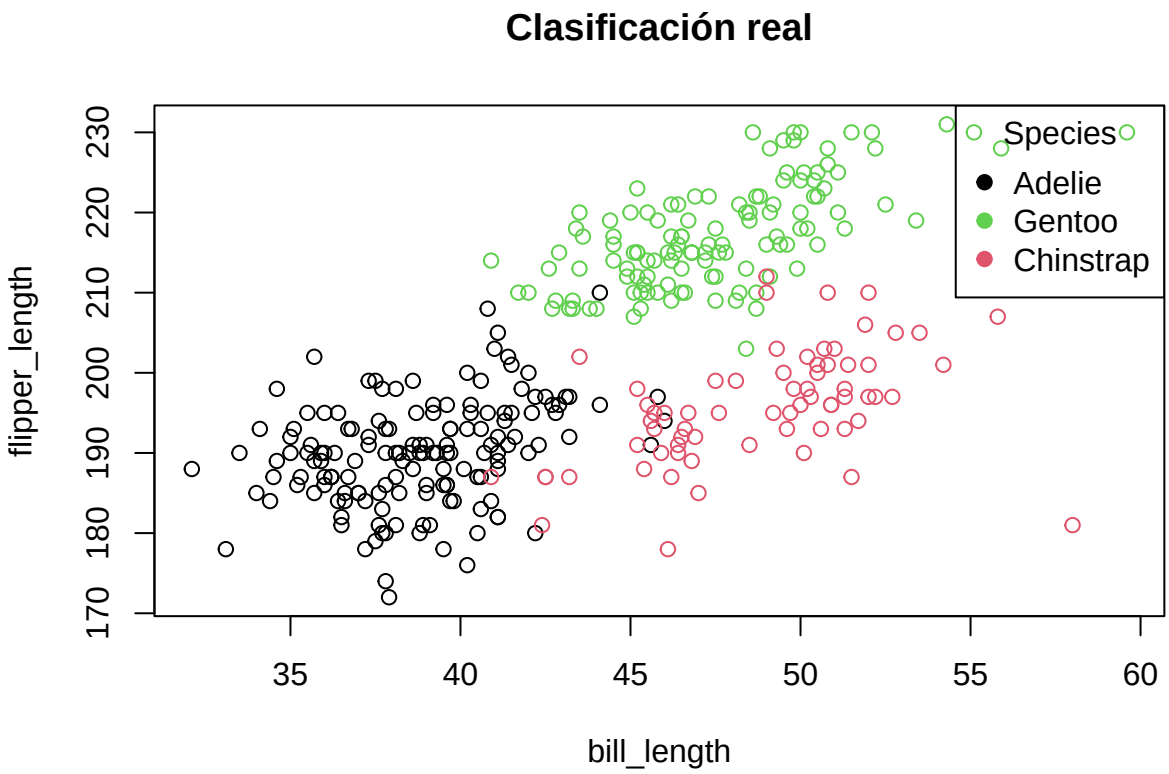

```r
# Plot with real species classification
plot(x[c(3,4)],
     col = as.factor(x_species$species), # Color by real species
     main = "Clasificación real",        # Plot title
     xlab = "flipper_length",
     ylab = "body_mass")

# Add a legend for species
legend("topright",
       legend = unique(x_species$species),        # Species labels
       col = unique(as.factor(x_species$species)), # Colors corresponding to species
       pch = 19,                                  # Point symbol in the legend
       title = "Species")
```




```
# bill_length y flipper_length
plot(x[c(1,3)],
     col = penguins3clusters$cluster,    # Color by k-means cluster
     main = "Clasificación k-means",     # Plot title
     xlab = "bill_length",
     ylab = "flipper_length")

# Add a legend for the clusters
legend("topright",
       legend = unique(penguins3clusters$cluster), # Cluster labels
       col = unique(penguins3clusters$cluster),    # Colors corresponding to clusters
       pch = 19,                                   # Point symbol in the legend
       title = "K-means Clusters")
```




```r
# Plot with real species classification
plot(x[c(1,3)],
     col = as.factor(x_species$species), # Color by real species
     main = "Clasificación real",        # Plot title
     xlab = "bill_length",
     ylab = "flipper_length")

# Add a legend for species
legend("topright",
       legend = unique(x_species$species),        # Species labels
       col = unique(as.factor(x_species$species)), # Colors corresponding to species
       pch = 19,                                  # Point symbol in the legend
       title = "Species")
```



Las dos medidas de *bill* parecen lograr mejores resultados al dividir las tres especies de pingüinos. El grupo formado por los puntos negros que ha encontrado el algoritmo coincide con los de la especie *Adelie*. Los otros dos grupos sin embargo se entremezclan algo más, y hay ciertos puntos que se clasifican como *Gentoo* (verde) cuando en realidad son *Chinstrap* (rojo).

Una buena técnica que ayuda a entender los grupos que se han formado, es mirar de darles un nombre. Cómo por ejemplo:

- Grupo 1: Sólo *Adelie* (color negro)
- Grupo 2: Principalmente *Chinstrap* (color rojo)
- Grupo 3: Mezcla de *Gentoo* (color verde) y *Adelie* (color negro)

Esto nos ayuda a entender cómo están formados los grupos y a referirnos a ellos en análisis posteriores.

Todo esto nos indica que el número de grupos o clúsers en un juego de datos no es un aspecto que podamos asegurar que siempre vamos a encontrar de forma precisa y objetiva, bien al contrario es un ámbito que requiere de análisis en sí mismo.

Os compartimos en el siguiente enlace un material didáctico complementario que os puede ayudar a profundizar en el tema de la selección del número de clústers más adecuado para un juego de datos: datascience.recursos.uoc.edu

Como continuación del estudio podríamos seguir experimentando combinando en gráficos similares a los anteriores. En definitiva se trataría en este punto de profundizar más en el conocimiento de las propiedades de las diferentes características o columnas del juego de datos.

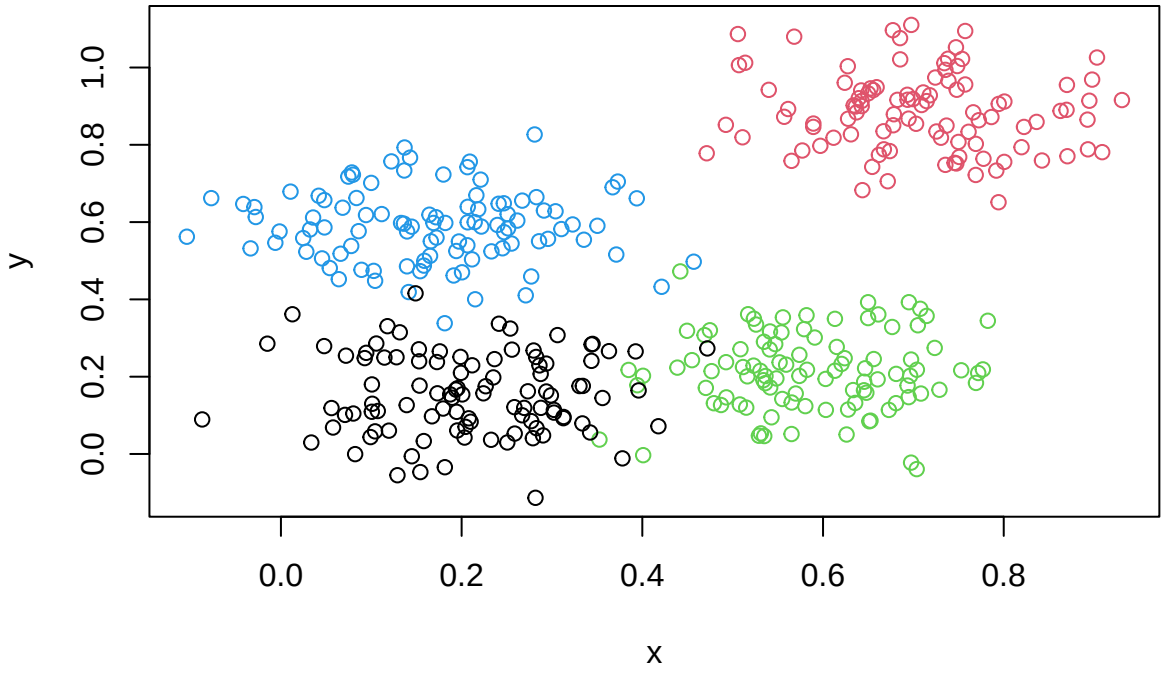
Ejemplo guiado 2

Métodos basados en densidad: DBSCAN y OPTICS

En este ejemplo vamos ha trabajar los algoritmos **DBSCAN** y **OPTICS** como métodos de clustering que permiten la generación de grupos no radiales a diferencia de k-means. Veremos que su parámetro de entrada más relevante es *minPts* que define la mínima densidad aceptada alrededor de un centroide.

Incrementar este parámetro nos permitirá reducir el ruido (observaciones no asignadas a ningún cluster), en cualquier caso empezaremos por construir nuestro propio juego de datos en el que dibujaremos 4 zonas de puntos diferenciadas.

```
if (!require('dbscan')) install.packages('dbscan')
library(dbscan)
set.seed(2)
n <- 400
x <- cbind(
x = runif(4, 0, 1) + rnorm(n, sd=0.1),
y = runif(4, 0, 1) + rnorm(n, sd=0.1)
)
plot(x, col=rep(1:4, time = 100))
```



Una de las primeras actividades que realiza el algoritmo es **ordenar las observaciones** de forma que los puntos más cercanos se conviertan en vecinos en el ordenamiento. Se podría pensar como una representación numérica del dendograma de una agrupación jerárquica.

```
### Lanzamos el algoritmo OPTICS dejando el parámetro eps con su valor por defecto y fijando el criterio
res <- optics(x, minPts = 10)
res
```

```
## OPTICS ordering/clustering for 400 objects.
## Parameters: minPts = 10, eps = 0.193786846197958, eps_cl = NA, xi = NA
## Available fields: order, reachdist, coredist, predecessor, minPts, eps,
##                   eps_cl, xi
```

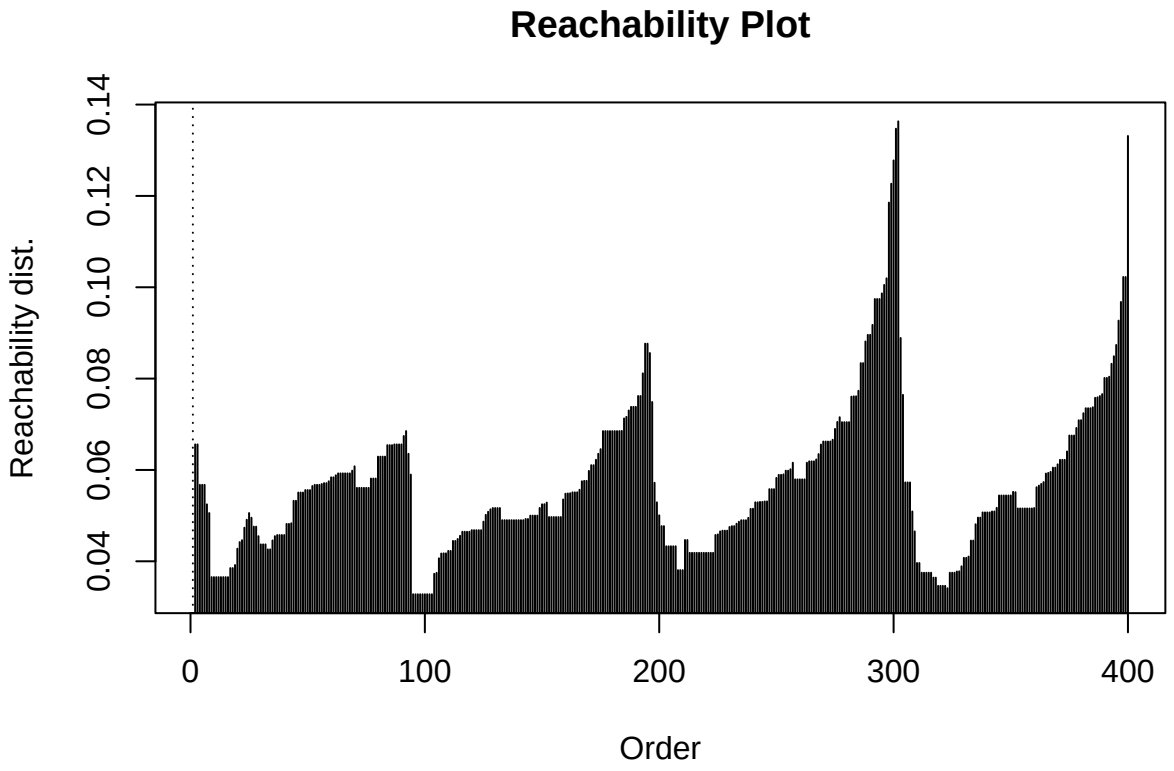
```
### Obtenemos la ordenación de las observaciones o puntos
res$order
```

```
##   [1]   1 363 209 349 337 301 357 333 321 285 281 253 241 177 153  57 257  29
##  [19]  77 169 105 293 229 145 181 385 393 377 317 381 185 117 101   9  73 237
##  [37] 397 369 365 273 305 245 249 309 157 345 213 205  97  49  33  41 193 149
##  [55]  17  83 389  25 121 329   5 161 341 217 189 141  85  53 225 313 289 261
##  [73] 221 173  69  61 297 125  81 133 129 197 109 137  59  93 165  89  21  13
##  [91] 277 191 203 379 399 375 351 311 235 231 227  71  11 299 271 291 147  55
## [109]  23 323 219 275  47 263   3 367 331 175  87 339 319 251 247 171 111 223
## [127]  51  63 343 303 207 151 391 359 287 283 215 143 131 115  99  31 183  43
## [145] 243 199  79  27 295  67 347 255 239 195 187 139 107  39 119 179 395 371
## [163] 201 123 159  91 211 355 103 327  95   7 167  35 267 155 387 383 335 315
## [181] 259 135  15 113 279 373   4 353 265 127  45  37  19 276 224 361 260 288
## [199] 336 368 348 292 268 252 120 108  96  88  32  16 340 156 388 372 356 332
## [217] 304 220 188 168 136 124  56 236  28 244 392 184  76 380 232 100 116 112
## [235] 256  72   8 280  64  52 208 172 152 148 360 352 192 160 144 284 216  48
## [253]  84  92  36  20 212 272 264 200 128  80 180 364 196  12 132  40 324 308
## [271] 176 164  68 316 312 384 300 344 328 248 204 140 296  24 320 228  60  44
## [289] 233  65 400 376 240 163 104 396 307  75  14 325 269 262 234 382 294 206
## [307] 198 374 310 362 318 386 358 330 278 210 298 282 122  98  34  26 174 142
## [325]  46   6  62 118 190 202 114 322 286  38 242 394 342 266 162 130  30 182
## [343]   2  74 314 290 246 194 170 126 158 378 350 254 226 214  70  18  10 366
## [361] 354 186 150  86 306 102 338 346 134 250 138  94  78 390 274  58  42 258
## [379]  66  90 146 370 222 218 326  82 110 270 334 178 166 398  22  50 238 106
## [397] 154 302 230  54
```

Otro paso muy interesante del algoritmo es la generación de un **diagrama de alcanzabilidad** o *reachability plot,* en el que se aprecia de una forma visual la distancia de alcanzabilidad de cada punto.

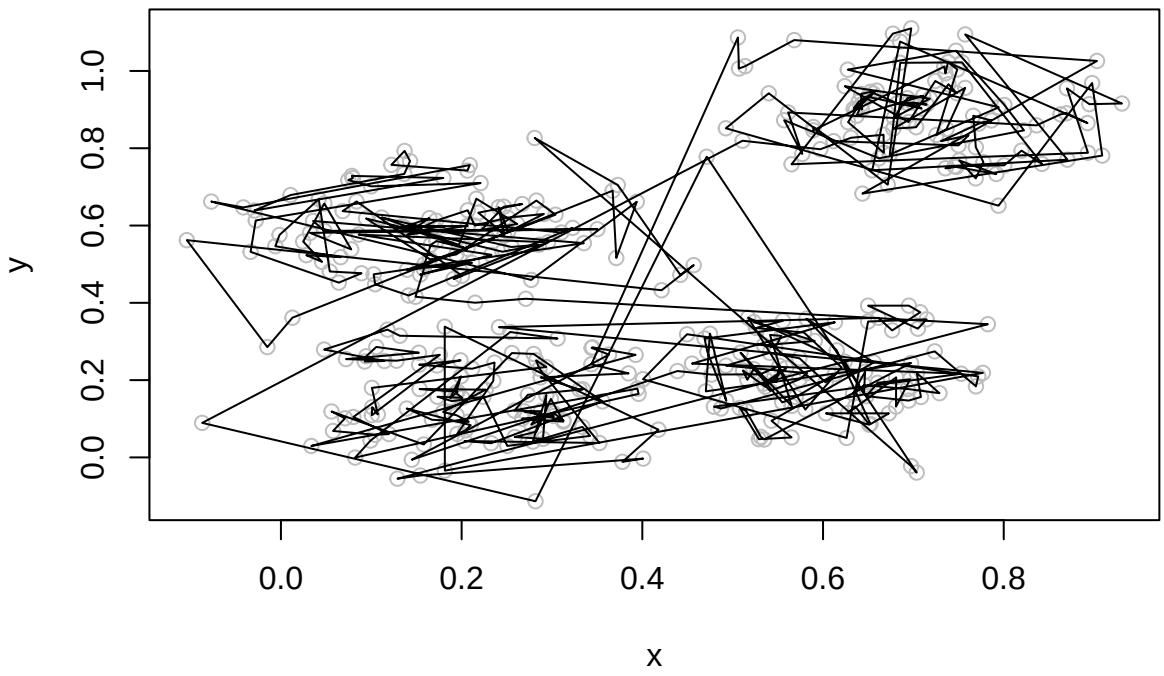
Los valles representan clusters (cuanto más profundo es el valle, más denso es el cluster), mientras que las cimas indican los puntos que están entre las agrupaciones (estos puntos son candidatos a ser considerados *outliers*)

```
### Gráfica de alcanzabilidad
plot(res)
```



Veamos otra representación del diagrama de alcanzabilidad, donde podemos observar las trazas de las distancias entre puntos cercanos del mismo cluster y entre clusters distintos.

```
### Dibujo de las trazas que relacionan puntos
plot(x, col = "grey")
polygon(x[res$order,])
```

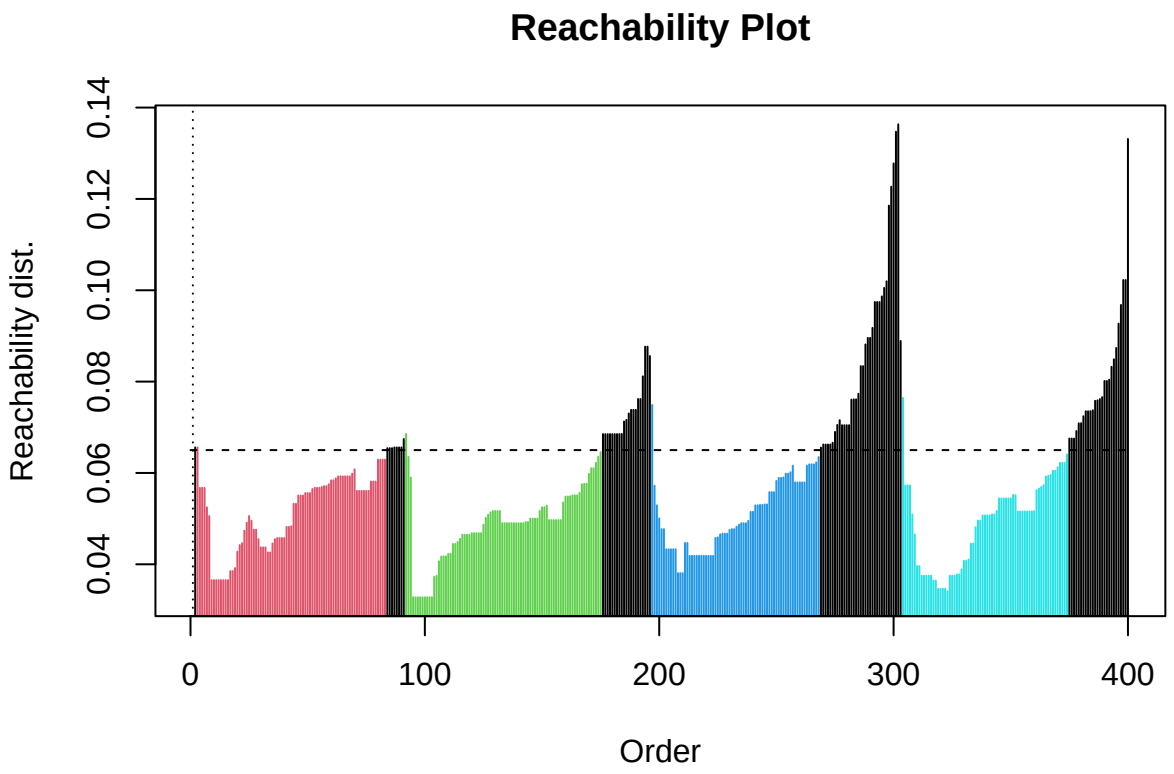


Otro ejercicio interesante a realizar es extraer una agrupación de la ordenación realizada por OPTICS similar a lo que DBSCAN hubiera generado estableciendo el parámetro eps en eps_cl = 0.065. En este sentido animamos al estudiante a experimentar con diferentes valores de este parámetro.

```
### Extracción de un clustering DBSCAN cortando la alcanzabilidad en el valor eps_cl
res <- extractDBSCAN(res, eps_cl = .065)
res
```

```
## OPTICS ordering/clustering for 400 objects.
## Parameters: minPts = 10, eps = 0.193786846197958, eps_cl = 0.065, xi = NA
## The clustering contains 4 cluster(s) and 92 noise points.
##
##  0  1  2  3  4
## 92 81 84 72 71
##
## Available fields: order, reachdist, coredist, predecessor, minPts, eps,
##                   eps_cl, xi, cluster
```

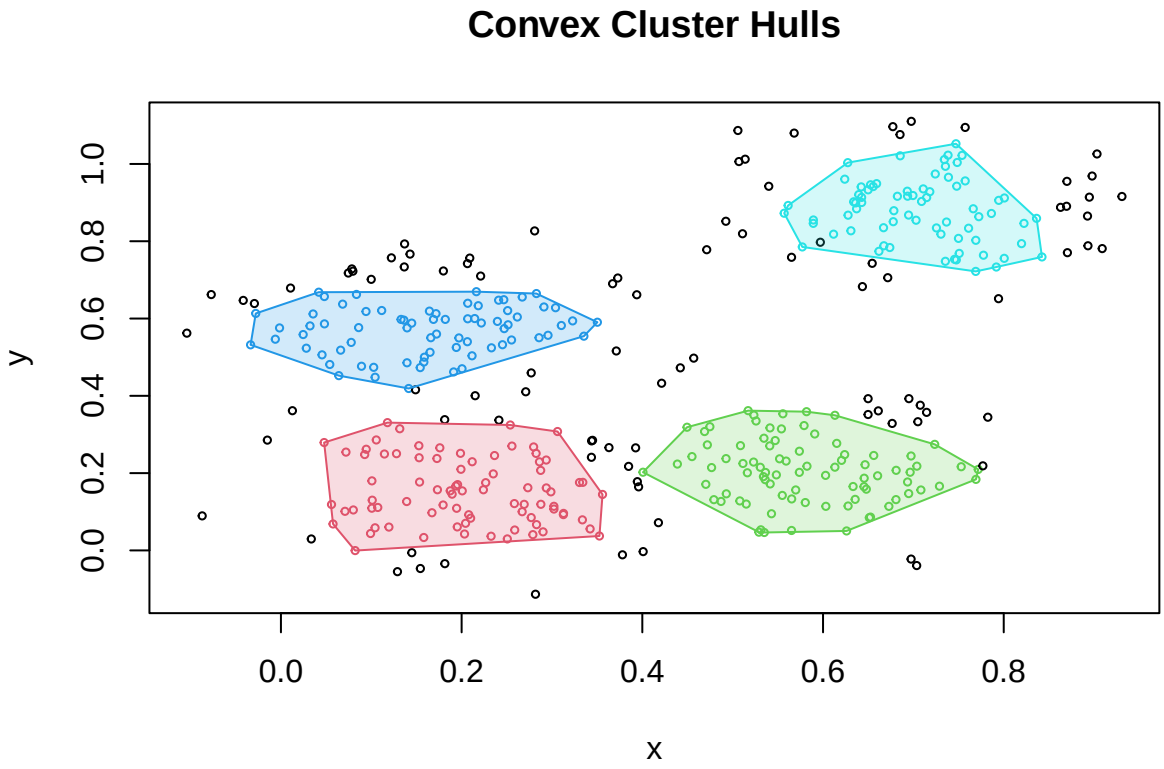
```
plot(res) ## negro indica ruido
```



Observamos en el gráfico anterior como se han coloreado los 4 clusters y en negro se mantienen los valores *outliers* o extremos.

Seguimos adelante con una representación gráfica que nos muestra los clusters mediante formas convexas.

```
hullplot(x, res)
```

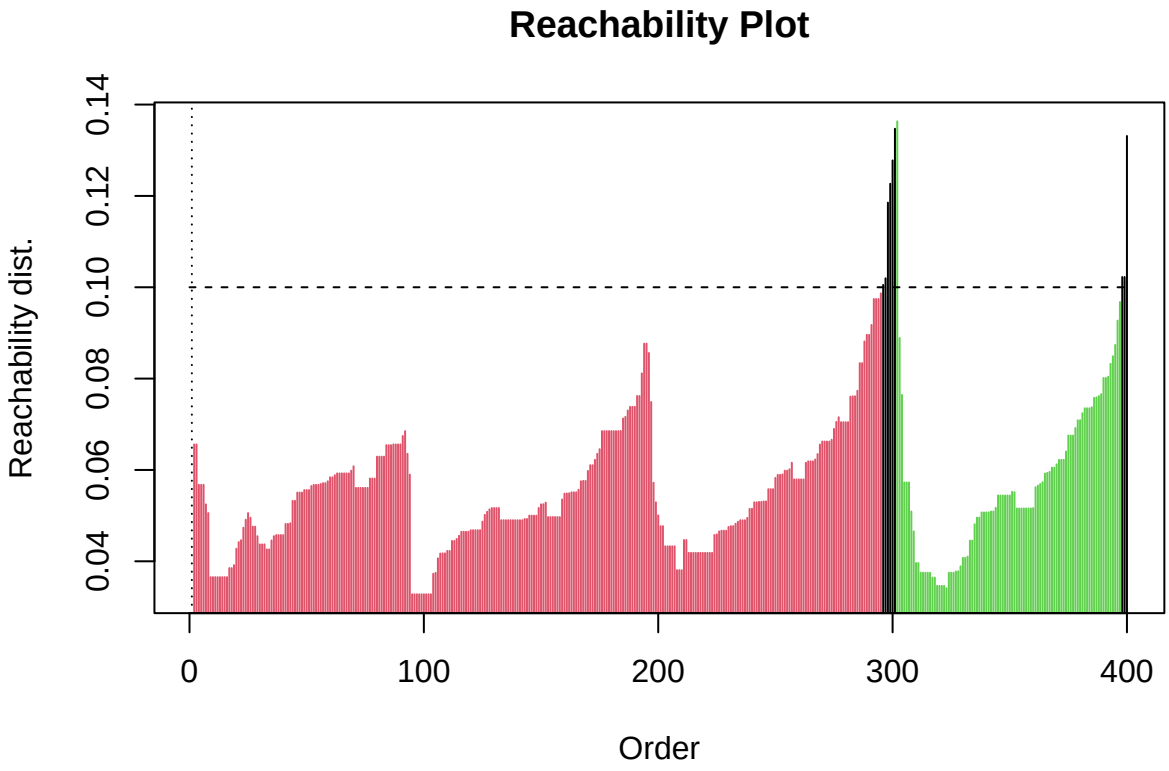


Repetimos el experimento anterior incrementando el parámetro *epc_c*, veamos como el efecto que produce es la concentración de clusters ya que flexibilizamos la condición de densidad.

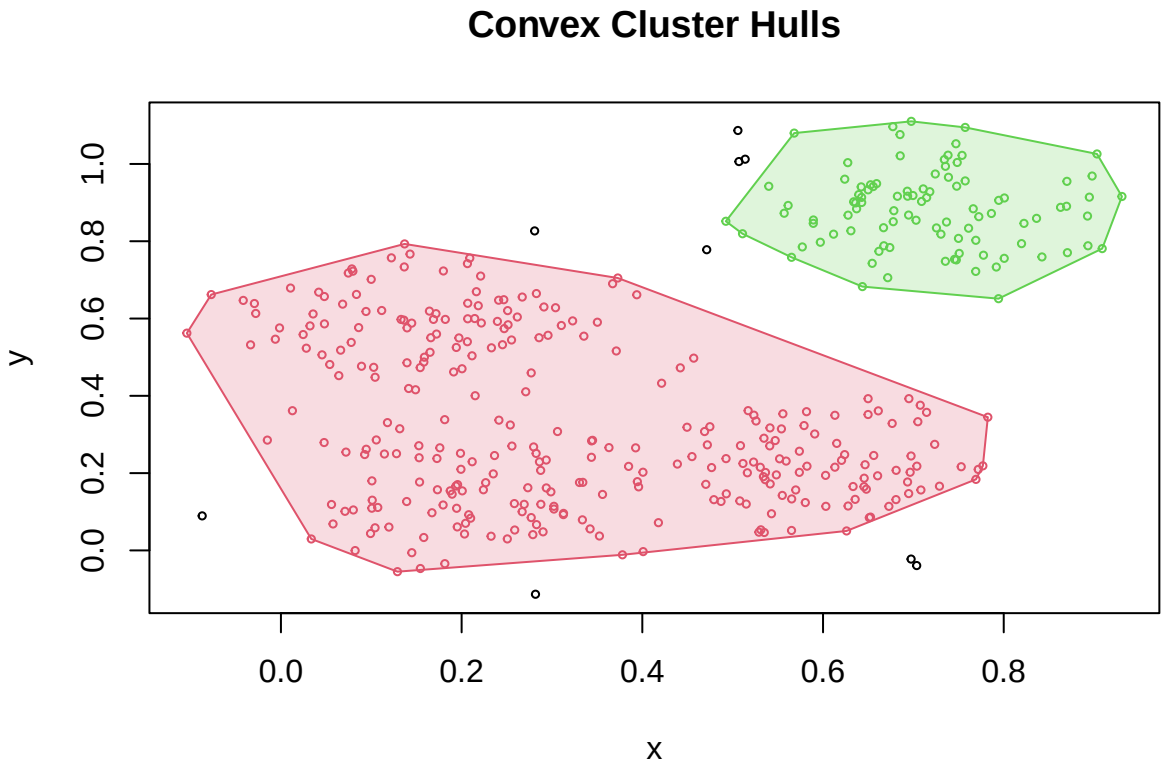
```
### Incrementamos el parámetro eps
res <- extractDBSCAN(res, eps_cl = .1)
res
```

```
## OPTICS ordering/clustering for 400 objects.
## Parameters: minPts = 10, eps = 0.193786846197958, eps_cl = 0.1, xi = NA
## The clustering contains 2 cluster(s) and 9 noise points.
##
##   0   1   2
##   9 295  96
##
## Available fields: order, reachdist, coredist, predecessor, minPts, eps,
##                   eps_cl, xi, cluster
```

```
plot(res)
```




```
hullplot(x, res)
```

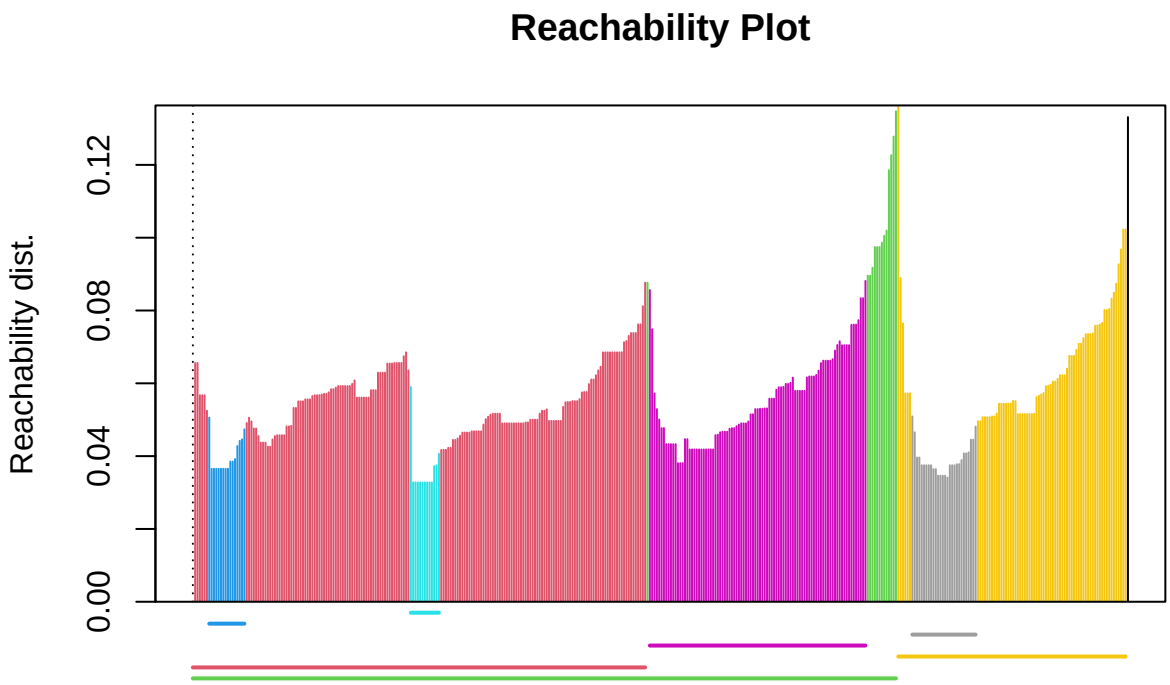


Veamos ahora una variante de la extracción **DBSCN** anterior. En ella el parámetro *xi* nos va a servir para clasificar los clusters en función del cambio en la densidad relativa de los mismos.

```
### Extracción del clustering jerárquico en función de la variación de la densidad por el método xi
res <- extractXi(res, xi = 0.05)
res
```

```
## OPTICS ordering/clustering for 400 objects.
## Parameters: minPts = 10, eps = 0.193786846197958, eps_cl = NA, xi = 0.05
## The clustering contains 7 cluster(s) and 1 noise points.
##
## Available fields: order, reachdist, coredist, predecessor, minPts, eps,
##                   eps_cl, xi, cluster, clusters_xi
```

```
plot(res)
```




```
x
```

```
##                x           y
##   [1,] 0.343666793 0.240999810
##   [2,] 0.589336469 0.854988778
##   [3,] 0.565301159 0.133186578
##   [4,] 0.181293949 0.338308761
##   [5,] 0.255677733 0.270028919
##   [6,] 0.678404234 0.879046494
##   [7,] 0.771773728 0.209604522
##   [8,] 0.154173219 0.473024576
##   [9,] 0.226647335 0.175297344
##  [10,] 0.800549314 0.755782023
##  [11,] 0.534056799 0.190858423
##  [12,] 0.064085023 0.452245791
##  [13,] 0.363105156 0.265995416
##  [14,] 0.471267127 0.778066037
##  [15,] 0.661186793 0.361091147
##  [16,] 0.171632592 0.612627706
##  [17,] 0.286165129 0.229181911
##  [18,] 0.745600551 0.753282342
##  [19,] 0.782408255 0.344706276
##  [20,] 0.048059338 0.656877109
##  [21,] 0.343846080 0.283401065
##  [22,] 0.897839200 0.968719839
##  [23,] 0.573820113 0.256622399
```

```
##  [24,] -0.077118718  0.6619866818
##  [25,]  0.232605990  0.0367075603
##  [26,]  0.642718219  0.9134394632
##  [27,]  0.652546662  0.0862235414
##  [28,]  0.197015591  0.5498234623
##  [29,]  0.258776120  0.0528098630
##  [30,]  0.734270076  1.0113198620
##  [31,]  0.680942770  0.2072583216
##  [32,]  0.139636148  0.5757941299
##  [33,]  0.107214733  0.1111767020
##  [34,]  0.642807986  0.9002650688
##  [35,]  0.400728357  0.2023300691
##  [36,]  0.077793472  0.5382192377
##  [37,]  0.128976068 -0.0549563151
##  [38,]  0.677722779  0.8505778612
##  [39,]  0.534967712  0.0464137329
##  [40,] -0.027858397  0.6131642651
##  [41,]  0.100711754  0.1799650475
##  [42,]  0.892728783  0.8651419522
##  [43,]  0.635575728  0.1320770866
##  [44,]  0.367143964  0.6900697974
##  [45,]  0.154333887 -0.0468030070
##  [46,]  0.693289612  0.9294418142
##  [47,]  0.554910190  0.1422907173
##  [48,]  0.048175144  0.5861547699
##  [49,]  0.101053545  0.1299186073
##  [50,]  0.909004172  0.7809617852
##  [51,]  0.517101629  0.3616790558
##  [52,]  0.295623472  0.5566242202
##  [53,]  0.080124997  0.1046771207
##  [54,]  0.505786212  1.0865660065
##  [55,]  0.541029225  0.2705037183
##  [56,]  0.261638173  0.6039368564
##  [57,]  0.298805240  0.1515136056
##  [58,]  0.869535913  0.8904404900
##  [59,]  0.394502114  0.1779099857
##  [60,]  0.371176172  0.5160687801
##  [61,]  0.114567827  0.2494316280
##  [62,]  0.718190512  0.9281434335
##  [63,]  0.623949815  0.2478928682
##  [64,]  0.086052410  0.5765330633
##  [65,] -0.015002440  0.2854203396
##  [66,]  0.654444777  0.7429071201
##  [67,]  0.581744325  0.3589483296
##  [68,]  0.078503259  0.7283786431
##  [69,]  0.092754693  0.2480560308
##  [70,]  0.735418986  0.7481143885
##  [71,]  0.559160254  0.2311712332
##  [72,]  0.211536697  0.5036108973
##  [73,]  0.179509997  0.1175372140
##  [74,]  0.611662998  0.8182782994
##  [75,]  0.703677558 -0.0392058861
##  [76,]  0.245230898  0.5322250503
##  [77,]  0.290134819  0.0480401854
```

```
##  [78,]  0.561370202  0.8925764008
##  [79,]  0.672924794  0.1140114350
##  [80,] -0.001524570  0.5756718220
##  [81,]  0.131545046  0.3148125484
##  [82,]  0.565147091  0.7584371526
##  [83,]  0.352534357  0.0372840465
##  [84,]  0.350264172  0.5904631187
##  [85,]  0.119542919  0.0604101445
##  [86,]  0.673905914  0.7837206509
##  [87,]  0.534631374  0.2903171487
##  [88,]  0.206721418  0.5993830761
##  [89,]  0.344921345  0.2848366485
##  [90,]  0.870489532  0.7703980758
##  [91,]  0.454965696  0.2427886526
##  [92,]  0.032206195  0.5809249131
##  [93,]  0.033615180  0.0296248067
##  [94,]  0.577063546  0.7852711633
##  [95,]  0.769262043  0.1838425799
##  [96,]  0.168816508  0.5977552716
##  [97,]  0.100620740  0.1092499776
##  [98,]  0.642258025  0.9406324015
##  [99,]  0.680772275  0.1314689274
## [100,]  0.194111704  0.5251772098
## [101,]  0.153455062  0.1768613704
## [102,]  0.627411026  1.0033389153
## [103,]  0.487106502  0.1267209645
## [104,]  0.372855951  0.7048347170
## [105,]  0.278874268  0.0407955944
## [106,]  0.903242748  1.0258574963
## [107,]  0.531188978  0.0532687121
## [108,]  0.132968478  0.5977063582
## [109,]  0.082144200 -0.0004108927
## [110,]  0.677322123  1.0964817431
## [111,]  0.620512281  0.2330296431
## [112,]  0.303945902  0.6279811561
## [113,]  0.241299120  0.3370488256
## [114,]  0.747972045  0.9424109289
## [115,]  0.696421701  0.2022852242
## [116,]  0.282765605  0.6646255995
## [117,]  0.195542064  0.1702358431
## [118,]  0.624042370  0.9605528173
## [119,]  0.697446318  0.2443465542
## [120,]  0.181937762  0.5977895548
## [121,]  0.355945419  0.1448650182
## [122,]  0.659309938  0.9490330550
## [123,]  0.468903377  0.3075845768
## [124,]  0.221809873  0.5884413871
## [125,]  0.117923661  0.3306591875
## [126,]  0.766254597  0.8840610731
## [127,]  0.400927351 -0.0031195882
## [128,] -0.006191088  0.5464662592
## [129,]  0.253862677  0.3245371894
## [130,]  0.735470354  0.9938707544
## [131,]  0.660433106  0.1930497815
```

```
## [132,] -0.033572638  0.5319595549
## [133,]  0.306140170  0.3075964593
## [134,]  0.822423506  0.8462520414
## [135,]  0.676533167  0.3285949180
## [136,]  0.246692946  0.6486607727
## [137,]  0.395889611  0.1644393277
## [138,]  0.556993051  0.8724565785
## [139,]  0.515015950  0.1199066711
## [140,]  0.209024319  0.7564028556
## [141,]  0.104184096  0.0585206179
## [142,]  0.710929080  0.9354756819
## [143,]  0.647950652  0.1584490530
## [144,]  0.102684614  0.4737402900
## [145,]  0.250592858  0.0297681010
## [146,]  0.757364959  1.0945076835
## [147,]  0.492653399  0.2375312736
## [148,]  0.068313949  0.6372306759
## [149,]  0.282471324  0.2510692409
## [150,]  0.685431718  1.0210177721
## [151,]  0.645545513  0.1868850582
## [152,]  0.083610060  0.6623527059
## [153,]  0.312611628  0.0926032533
## [154,]  0.568062981  1.0797851509
## [155,]  0.649860402  0.3516365167
## [156,]  0.214472177  0.6000032921
## [157,]  0.211681588  0.2294642595
## [158,]  0.769126305  0.8023929905
## [159,]  0.613173063  0.3496562888
## [160,]  0.104244817  0.4481187653
## [161,]  0.158110970  0.0334433725
## [162,]  0.738361992  1.0226917316
## [163,]  0.442039725  0.4724116329
## [164,]  0.079654959  0.7225391321
## [165,]  0.392591739  0.2654213383
## [166,]  0.492451473  0.8516692647
## [167,]  0.449475738  0.3187349141
## [168,]  0.267095229  0.6555618860
## [169,]  0.293748446  0.1616841309
## [170,]  0.786359261  0.8718688845
## [171,]  0.579012199  0.3229909276
## [172,]  0.200439725  0.4700693004
## [173,]  0.094415393  0.2617130863
## [174,]  0.637155651  0.8839538914
## [175,]  0.547080871  0.2842904060
## [176,]  0.074585636  0.7177169440
## [177,]  0.266998381  0.1001620518
## [178,]  0.539948119  0.9422123102
## [179,]  0.470285968  0.1710284054
## [180,]  0.041858989  0.6681429228
## [181,]  0.224100723  0.1569330588
## [182,]  0.589230210  0.8462875284
## [183,]  0.627740783  0.1149317490
## [184,]  0.285712814  0.5503887006
## [185,]  0.187405117  0.1541767308
```

```
## [186,]  0.753887353  1.0221753006
## [187,]  0.507915359  0.1280700986
## [188,]  0.218416119  0.6331477664
## [189,]  0.057670338  0.0684327225
## [190,]  0.694696921  0.8676847865
## [191,]  0.438794397  0.2234585140
## [192,]  0.141420164  0.4189815614
## [193,]  0.293638559  0.2338280706
## [194,]  0.772430815  0.8635468302
## [195,]  0.529050383  0.0472099902
## [196,]  0.089199924  0.4765599101
## [197,]  0.099204689  0.0437217372
## [198,]  0.627732134  0.8674179541
## [199,]  0.603124697  0.1140490257
## [200,]  0.066096701  0.5181195136
## [201,]  0.471972002  0.2731005482
## [202,]  0.724245037  0.9739914424
## [203,]  0.476670902  0.2143021976
## [204,]  0.206435739  0.7421071431
## [205,]  0.172692653  0.2378389903
## [206,]  0.667319279  0.7882971117
## [207,]  0.633272697  0.1653715642
## [208,]  0.191179328  0.4618303115
## [209,]  0.287506880  0.2071157453
## [210,]  0.649929151  0.9330066992
## [211,]  0.752946959  0.2165208943
## [212,]  0.024632280  0.5585863014
## [213,]  0.198750877  0.2512165469
## [214,]  0.747520409  0.7516802060
## [215,]  0.694803747  0.1470863459
## [216,]  0.035603808  0.6118437344
## [217,]  0.070919482  0.1011847759
## [218,]  0.869938893  0.9549930160
## [219,]  0.613140956  0.2153658404
## [220,]  0.239603834  0.5925878798
## [221,]  0.105562746  0.2858335585
## [222,]  0.894759201  0.9140417807
## [223,]  0.580370333  0.1238221354
## [224,]  0.214979924  0.4003191018
## [225,]  0.175897051  0.2654816040
## [226,]  0.791789199  0.7332688142
## [227,]  0.535955347  0.1831766341
## [228,]  0.393877111  0.6615353563
## [229,]  0.342192438  0.0556345734
## [230,]  0.506941863  1.0062023230
## [231,]  0.511394091  0.2249572935
## [232,]  0.165057148  0.5126890752
## [233,]  0.012716418  0.3615122603
## [234,]  0.597028356  0.7974191399
## [235,]  0.516260086  0.2008665678
## [236,]  0.216323360  0.6694478584
## [237,]  0.194474649  0.1090056177
## [238,]  0.643590057  0.6826494957
## [239,]  0.565148331  0.0516121890
```

```
## [240,]  0.456893746  0.4974903357
## [241,]  0.302164795  0.1070223371
## [242,]  0.757338358  0.9561099816
## [243,]  0.651236786  0.0845884398
## [244,]  0.111538795  0.6207407882
## [245,]  0.195177689  0.0608268396
## [246,]  0.730526187  0.8183257219
## [247,]  0.525883217  0.3348842737
## [248,]  0.122251500  0.7570261404
## [249,]  0.235036577  0.1980726892
## [250,]  0.661861098  0.7739562076
## [251,]  0.555587776  0.3535402699
## [252,]  0.136097463  0.5958859614
## [253,]  0.269227299  0.1192178054
## [254,]  0.750739701  0.7685767585
## [255,]  0.542959320  0.0949099416
## [256,]  0.139542311  0.4854989833
## [257,]  0.282925928  0.0667178532
## [258,]  0.685242089  1.0760407563
## [259,]  0.707627649  0.3756863215
## [260,]  0.158216077  0.4874397151
## [261,]  0.152682712  0.2709327982
## [262,]  0.510900586  0.8193948500
## [263,]  0.508622196  0.2710860617
## [264,]  0.028108236  0.5232488785
## [265,]  0.377999687 -0.0113662915
## [266,]  0.748531265  1.0038079546
## [267,]  0.626070209  0.0503973601
## [268,]  0.164382793  0.6191874183
## [269,] -0.087134543  0.0893511230
## [270,]  0.697661484  1.1102093763
## [271,]  0.573799982  0.2020141495
## [272,]  0.045521546  0.5060468532
## [273,]  0.209974781  0.0834127893
## [274,]  0.862894030  0.8877120211
## [275,]  0.569774347  0.1564543205
## [276,]  0.270993729  0.4104968969
## [277,]  0.144877968 -0.0058880731
## [278,]  0.709098357  0.9029481310
## [279,]  0.776907120  0.2188366583
## [280,]  0.233003801  0.5242105730
## [281,]  0.312817413  0.0957866266
## [282,]  0.639554102  0.9205999753
## [283,]  0.708359537  0.1563603441
## [284,]  0.335310918  0.5543347835
## [285,]  0.302155216  0.1141675676
## [286,]  0.702991930  0.8544127674
## [287,]  0.703877557  0.2179948899
## [288,]  0.158753237  0.4999378526
## [289,]  0.072007278  0.2543694628
## [290,]  0.761108674  0.8336767989
## [291,]  0.582242723  0.2178924730
## [292,]  0.144755478  0.5881348597
## [293,]  0.333659848  0.0790684416
```

```
## [294,]  0.666964710  0.8349756841
## [295,]  0.615191177  0.2770177211
## [296,] -0.041691634  0.6469100779
## [297,]  0.047834242  0.2791033528
## [298,]  0.633927750  0.9020294119
## [299,]  0.541514351  0.1720673748
## [300,]  0.136502895  0.7334987745
## [301,]  0.273314492  0.1621863570
## [302,]  0.513831905  1.0122431633
## [303,]  0.646544268  0.2217346601
## [304,]  0.247106387  0.5739207753
## [305,]  0.204367576  0.0701756761
## [306,]  0.794491835  0.9057870025
## [307,]  0.697471956 -0.0223887373
## [308,]  0.099934840  0.7014104530
## [309,]  0.236415306  0.2454380124
## [310,]  0.655552927  0.9415122575
## [311,]  0.523120603  0.2286062598
## [312,] -0.029347267  0.6387932949
## [313,]  0.127937670  0.2503043454
## [314,]  0.749907091  0.8073912159
## [315,]  0.695048104  0.3926817182
## [316,]  0.179820877  0.7230601358
## [317,]  0.193718931  0.1665678935
## [318,]  0.682312328  0.9163367877
## [319,]  0.523467979  0.3499618084
## [320,]  0.277018674  0.4593050984
## [321,]  0.277045338  0.0849633575
## [322,]  0.738829392  0.9655671278
## [323,]  0.602841150  0.1941826572
## [324,]  0.221046762  0.7098663577
## [325,]  0.281761240 -0.1133958605
## [326,]  0.893234610  0.7882655439
## [327,]  0.723605680  0.2744348054
## [328,]  0.136869126  0.7930084981
## [329,]  0.279684129  0.2677575611
## [330,]  0.693606760  0.9165625567
## [331,]  0.553953704  0.3144308923
## [332,]  0.240975993  0.6474136391
## [333,]  0.287722746  0.1193454409
## [334,]  0.671655002  0.7056706045
## [335,]  0.649900494  0.3926316032
## [336,]  0.165906847  0.5503659949
## [337,]  0.334135048  0.1761032555
## [338,]  0.747141843  1.0522879845
## [339,]  0.590824018  0.3012783992
## [340,]  0.206776042  0.6395441377
## [341,]  0.055992816  0.1188492063
## [342,]  0.725524719  0.8344490182
## [343,]  0.645341020  0.1648376303
## [344,]  0.142908728  0.7667265269
## [345,]  0.152982672  0.2397606165
## [346,]  0.800718378  0.9118975004
## [347,]  0.656150430  0.2456795419
```

```
## [348,]  0.206331307  0.5400228508
## [349,]  0.330448236  0.1757896779
## [350,]  0.769020087  0.7219815569
## [351,]  0.536325609  0.2017292426
## [352,]  0.322877044  0.5938831770
## [353,]  0.417808662  0.0717507060
## [354,]  0.842316493  0.7593806003
## [355,]  0.479201753  0.1312919083
## [356,]  0.251382005  0.5838297250
## [357,]  0.258370919  0.1211560273
## [358,]  0.714695853  0.9130389653
## [359,]  0.728862693  0.1661881838
## [360,]  0.310377729  0.5819613864
## [361,]  0.149006544  0.4156536520
## [362,]  0.652599714  0.9460753033
## [363,]  0.384827531  0.2174660707
## [364,]  0.054089292  0.4810852497
## [365,]  0.139209358  0.1264651509
## [366,]  0.819707393  0.7937682622
## [367,]  0.541307604  0.3169748791
## [368,]  0.172219246  0.5598427623
## [369,]  0.167183942  0.0974700422
## [370,]  0.930847798  0.9157591511
## [371,]  0.492965287  0.1464324782
## [372,]  0.291115310  0.6299026829
## [373,]  0.181437802 -0.0342689005
## [374,]  0.636129950  0.8992165342
## [375,]  0.548366317  0.1954134349
## [376,]  0.421363328  0.4325614468
## [377,]  0.200694190  0.1540783758
## [378,]  0.777577954  0.7638789091
## [379,]  0.530553934  0.2151510812
## [380,]  0.094177088  0.6182588282
## [381,]  0.199246996  0.2099222038
## [382,]  0.630871848  0.8270320973
## [383,]  0.704974901  0.3330252226
## [384,]  0.010625189  0.6788433612
## [385,]  0.189498368  0.1455593911
## [386,]  0.699519654  0.9182545547
## [387,]  0.714639215  0.3572342097
## [388,]  0.250971221  0.6204666308
## [389,]  0.203217468  0.0424667836
## [390,]  0.836329737  0.8592245214
## [391,]  0.693559248  0.1770662211
## [392,]  0.255235475  0.5445164203
## [393,]  0.173239577  0.1569703413
## [394,]  0.736656884  0.8497206810
## [395,]  0.474779201  0.3200913196
## [396,]  0.280823068  0.8267366333
## [397,]  0.207865245  0.0923245055
## [398,]  0.794231091  0.6514053456
## [399,]  0.551918776  0.2370048784
## [400,] -0.104129696  0.5621596035
```

Ejercicios

Los ejercicios se realizarán en base al juego de datos *Hawks* presente en el paquete R *Stat2Data*.

Los estudiantes y el profesorado del Cornell College en Mount Vernon, Iowa, recogieron datos durante muchos años en el mirador de halcones del lago MacBride, cerca de Iowa City, en el estado de Iowa. El conjunto de datos que analizamos aquí es un subconjunto del conjunto de datos original, utilizando sólo aquellas especies para las que había más de 10 observaciones. Los datos se recogieron en muestras aleatorias de tres especies diferentes de halcones: Colirrojo, Gavilán y Halcón de Cooper.

Hemos seleccionado este juego de datos por su parecido con el juego de datos *penguins* y por su potencial a la hora de aplicarle algoritmos de minería de datos no supervisados.

```
if (!require('Stat2Data')) install.packages('Stat2Data')
library(Stat2Data)
data("Hawks")
summary(Hawks)
```

```
##      Month             Day             Year       CaptureTime   ReleaseTime
##  Min.   : 8.000   Min.   : 1.00   Min.   :1992   11:35  : 14          :842
##  1st Qu.: 9.000   1st Qu.: 9.00   1st Qu.:1995   13:30  : 14   11:00  :  2
##  Median :10.000   Median :16.00   Median :1999   11:45  : 13   11:35  :  2
##  Mean   : 9.843   Mean   :15.74   Mean   :1998   12:10  : 13   12:05  :  2
##  3rd Qu.:10.000   3rd Qu.:23.00   3rd Qu.:2001   14:00  : 13   12:50  :  2
##  Max.   :11.000   Max.   :31.00   Max.   :2003   13:05  : 12   13:32  :  2
##                                                  (Other):829   (Other): 56
##       BandNumber  Species  Age     Sex          Wing            Weight
##            :  2   CH: 70   A:224    :576   Min.   : 37.2   Min.   :  56.0
##  1142-09240:  1   RT:577   I:684   F:174   1st Qu.:202.0   1st Qu.: 185.0
##  1142-09241:  1   SS:261           M:158   Median :370.0   Median : 970.0
##  1142-09242:  1                           Mean   :315.6   Mean   : 772.1
##  1142-18229:  1                           3rd Qu.:390.0   3rd Qu.:1120.0
##  1142-19209:  1                           Max.   :480.0   Max.   :2030.0
##  (Other)   :901                           NA's   :1       NA's   :10
##      Culmen          Hallux            Tail        StandardTail
##  Min.   : 8.6   Min.   :  9.50   Min.   :119.0   Min.   :115.0
##  1st Qu.:12.8   1st Qu.: 15.10   1st Qu.:160.0   1st Qu.:162.0
##  Median :25.5   Median : 29.40   Median :214.0   Median :215.0
##  Mean   :21.8   Mean   : 26.41   Mean   :198.8   Mean   :199.2
##  3rd Qu.:27.3   3rd Qu.: 31.40   3rd Qu.:225.0   3rd Qu.:226.0
##  Max.   :39.2   Max.   :341.40   Max.   :288.0   Max.   :335.0
##  NA's   :7      NA's   :6                        NA's   :337
##      Tarsus        WingPitFat        KeelFat           Crop
##  Min.   :24.70   Min.   :0.0000   Min.   :0.000   Min.   :0.0000
##  1st Qu.:55.60   1st Qu.:0.0000   1st Qu.:2.000   1st Qu.:0.0000
##  Median :79.30   Median :1.0000   Median :2.000   Median :0.0000
##  Mean   :71.95   Mean   :0.7922   Mean   :2.184   Mean   :0.2345
##  3rd Qu.:87.00   3rd Qu.:1.0000   3rd Qu.:3.000   3rd Qu.:0.2500
##  Max.   :94.00   Max.   :3.0000   Max.   :4.000   Max.   :5.0000
##  NA's   :833     NA's   :831      NA's   :341     NA's   :343
```

Ejercicio 1

Presenta el juego de datos, nombre y significado de cada columna, así como las distribuciones de sus valores.

Realiza un estudio aplicando el método K-means, similar al de los ejemplos 1.1 y 1.2

Respuesta 1

```
data("Hawks")

# Exploramos la estructura del conjunto de datos
str(Hawks)
```

```
## 'data.frame':    908 obs. of  19 variables:
##  $ Month       : int  9 9 9 9 9 9 9 9 9 9 ...
##  $ Day         : int  19 22 23 23 27 28 28 29 29 30 ...
##  $ Year        : int  1992 1992 1992 1992 1992 1992 1992 1992 1992 1992 ...
##  $ CaptureTime : Factor w/ 308 levels " ","1:15","1:31",..: 181 25 138 42 62 71 181 88 261 192 ...
##  $ ReleaseTime : Factor w/ 60 levels "","  ","10:20",..: 1 2 2 2 2 2 2 2 2 2 ...
##  $ BandNumber  : Factor w/ 907 levels " ","1142-09240",..: 856 857 858 809 437 280 859 860 861 281 .
##  $ Species     : Factor w/ 3 levels "CH","RT","SS": 2 2 2 1 3 2 2 2 2 2 ...
##  $ Age         : Factor w/ 2 levels "A","I": 2 2 2 2 2 2 2 1 1 2 ...
##  $ Sex         : Factor w/ 3 levels "","F","M": 1 1 1 2 2 1 1 1 1 1 ...
##  $ Wing        : num  385 376 381 265 205 412 370 375 412 405 ...
##  $ Weight      : int  920 930 990 470 170 1090 960 855 1210 1120 ...
##  $ Culmen      : num  25.7 NA 26.7 18.7 12.5 28.5 25.3 27.2 29.3 26 ...
##  $ Hallux      : num  30.1 NA 31.3 23.5 14.3 32.2 30.1 30 31.3 30.2 ...
##  $ Tail        : int  219 221 235 220 157 230 212 243 210 238 ...
##  $ StandardTail: int  NA NA NA NA NA NA NA NA NA NA ...
##  $ Tarsus      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ WingPitFat  : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ KeelFat     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ Crop        : num  NA NA NA NA NA NA NA NA NA NA ...
```

Explicación, análisis y descripción del dataset El dataset es, tal y como se ha explicado en el enunciado, de un mirador de halcones. Son observaciones de estos entre 1992 y 2003.

Además son 908 registros y 19 variables, de las cuales 2 están casi vacías (Tarsus y WingPitFat) y 4 están con una cantidad importante con valores faltantes (Sex, StandardTail, KeelFat y Crop).

El motivo de que haya tantos valores faltantes puede ser de la diferencia de tiempo (11 años) en los cuales los criterios de regocida de datos hayan cambiado, así como dificultades para tomar el valor para algunos halcones, etc. Sería interesante ver si cambió el criterio o es aleatorio.

En líneas generales, el diccionario de datos del dataset es:

Month: Numérica, mes de la captura, siendo 8,9, 10 u 11 los meses que se hicieron capturas.

Day: Numérica, día en el que se capturó (entero del 1 al 31)

Year: Numérica, año de la captura

CaptureTime: Categórica, hora de captura

ReleaseTime: Categórica, hora de liberación

BandNumber: Categórica, ID de la banda asignada a cada halcón

Species: Categórica, especie del halcón; CH, RT, SS

Age: Categórica, edad del halcón; A (adult), I (inmaduro, inmmadure)

Sex: Categórica, sexo; F,M

Wing: Numérica, longitud alasen cm

Weight: Numérica, peso en gm

Hallux: Numérica, longitud de la garra

Culmen: Numérica, longitud del pico

Tail: Numérica, longitud de la cola

StandardTail: Numérica, medida alternativa a Tail con otra manera de medicción. Hay pocos registros que tienen valores.

Tarsus: Numérica, longitud del tarso

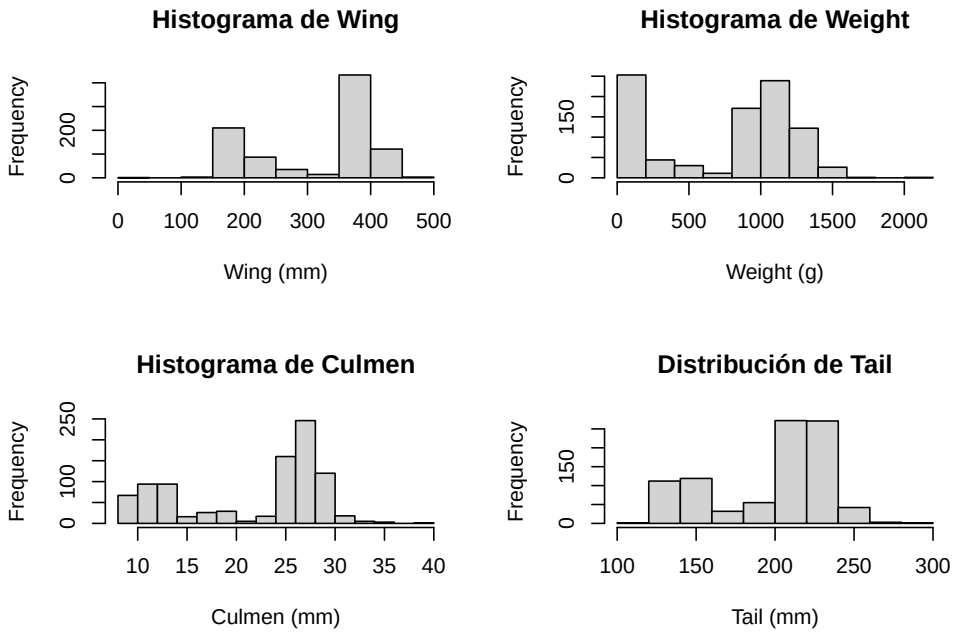
WingPitFat: Numérica, grasa acomulada en el ala

KeelFat: Numérica, grasa acomulada en el esternón

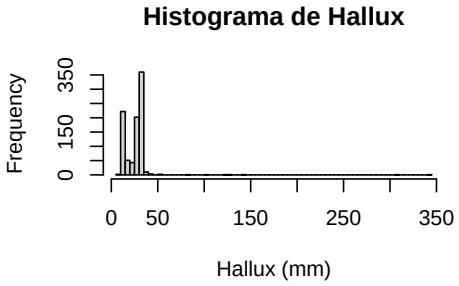
Crop: Numérica, cantidad de alimento en el buche

```
par(mfrow = c(2, 2))
hist(Hawks$Wing, main = "Histograma de Wing", xlab =  "Wing (mm)" )
hist(Hawks$Weight, main = "Histograma de Weight", xlab = "Weight (g)" )
hist(Hawks$Culmen, main = "Histograma de Culmen", xlab = "Culmen (mm)")
hist(Hawks$Tail, main = "Distribución de Tail", xlab = "Tail (mm)")
```

Distribucion de las variables. Histogramas



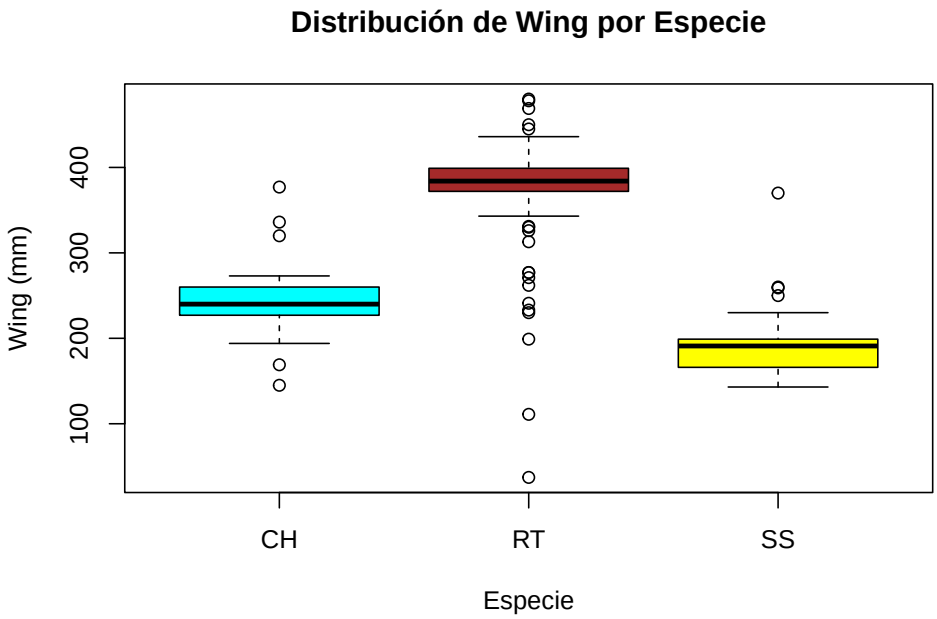
```r
#Hallux tiene un gran outlier, entonces tengo que poner muchos bins o breaks para que se vea la distribu
hist(Hawks$Hallux, main = "Histograma de Hallux", xlab = "Hallux (mm)", breaks = 100)
```



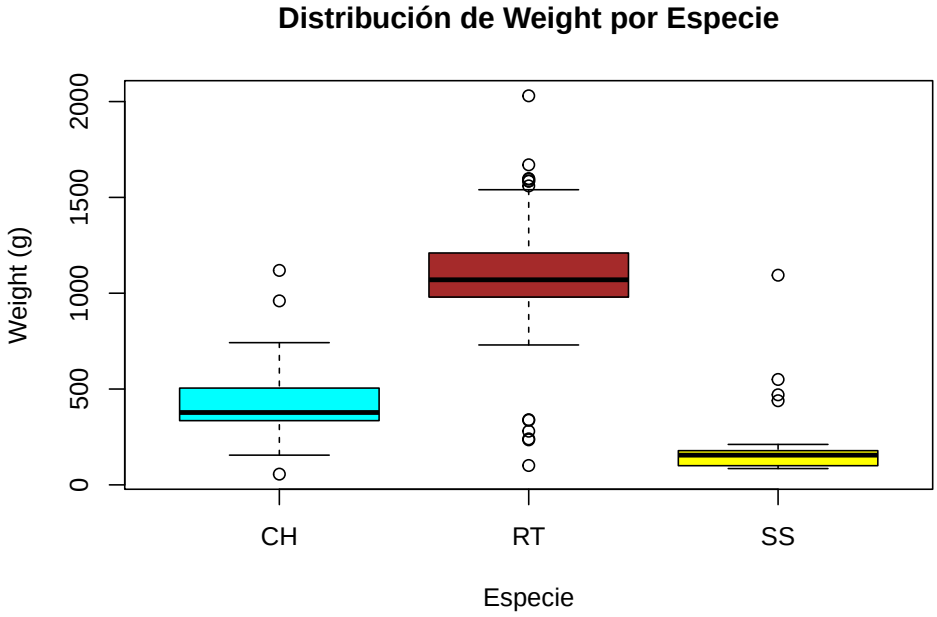
Bueno aqui vemos la distribución de los distintos valores, podemos ver que, no hay un patrón claro, y lo más llamativo es los outliers en Hallux.

```r
boxplot(Wing ~ Species, data = Hawks, main = "Distribución de Wing por Especie",
        ylab = "Wing (mm)", xlab = "Especie", col = c("cyan", "brown", "yellow"))
```

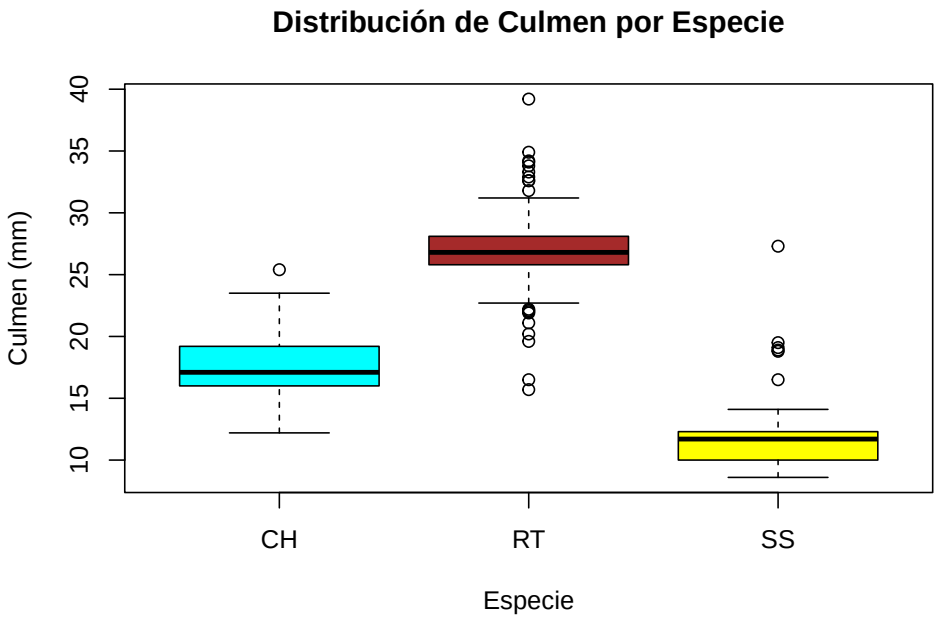
Box plot by especie.



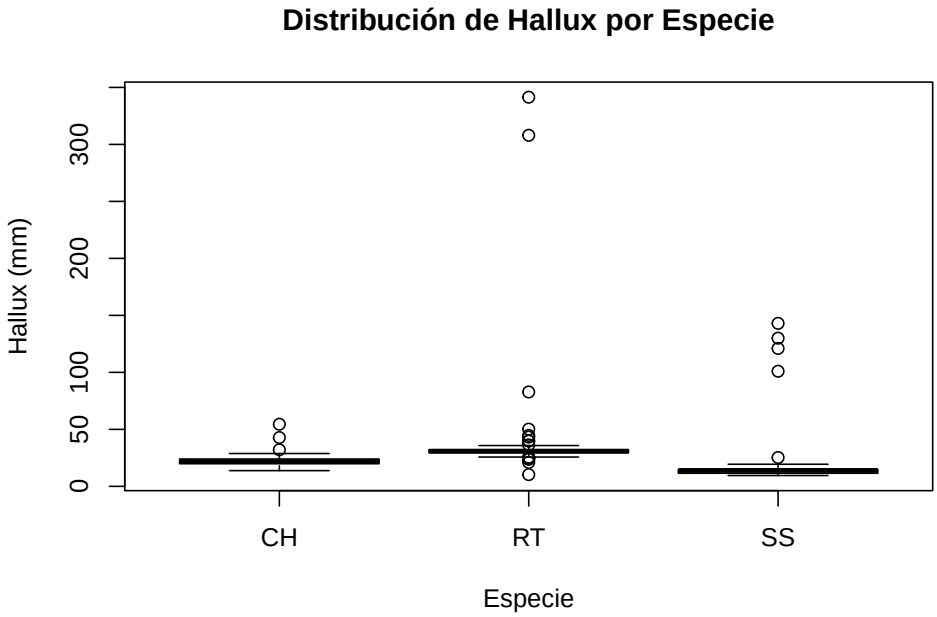

```
boxplot(Weight ~ Species, data = Hawks, main = "Distribución de Weight por Especie",
        ylab = "Weight (g)", xlab = "Especie", col = c("cyan", "brown", "yellow"))
```




```
boxplot(Culmen ~ Species, data = Hawks, main = "Distribución de Culmen por Especie",
        ylab = "Culmen (mm)", xlab = "Especie", col = c("cyan", "brown", "yellow"))
```




```
boxplot(Hallux ~ Species, data = Hawks, main = "Distribución de Hallux por Especie",
        ylab = "Hallux (mm)", xlab = "Especie", col = c("cyan", "brown", "yellow"))
```




```
hawks_subset <- subset(Hawks, Species %in% c("CH"))
summary(hawks_subset)
```

Box plots y summary para las especies CH y SS

```
##      Month             Day             Year       CaptureTime  ReleaseTime
##  Min.   : 9.000   Min.   : 1.00   Min.   :1992   10:45  : 2          :65
##  1st Qu.: 9.000   1st Qu.: 6.00   1st Qu.:1997   11:00  : 2   12:01  : 1
##  Median :10.000   Median :17.00   Median :2000   12:05  : 2   12:09  : 1
##  Mean   : 9.629   Mean   :15.67   Mean   :1999   12:20  : 2   14:07  : 1
##  3rd Qu.:10.000   3rd Qu.:23.75   3rd Qu.:2002   13:15  : 2   14:45  : 1
##  Max.   :11.000   Max.   :31.00   Max.   :2003   13:45  : 2   15:00  : 1
##                                                  (Other):58   (Other): 0
##       BandNumber Species Age    Sex         Wing           Weight
##            : 1   CH:70   A:32    : 2   Min.   :145.0   Min.   :  56.0
##  1204-45804: 1   RT: 0   I:38   F:35   1st Qu.:227.0   1st Qu.: 335.0
##  1204-45805: 1   SS: 0          M:33   Median :240.0   Median : 377.5
##  1204-45806: 1                         Mean   :244.1   Mean   : 420.5
##  1204-45809: 1                         3rd Qu.:260.0   3rd Qu.: 505.0
##  1204-45867: 1                         Max.   :377.0   Max.   :1119.0
##  (Other)   :64                          NA's   :1
##      Culmen          Hallux           Tail        StandardTail
##  Min.   :12.20   Min.   :13.80   Min.   :157.0   Min.   :181.0
##  1st Qu.:16.02   1st Qu.:19.93   1st Qu.:186.2   1st Qu.:196.0
##  Median :17.10   Median :21.60   Median :199.0   Median :206.0
##  Mean   :17.57   Mean   :22.82   Mean   :200.9   Mean   :206.9
##  3rd Qu.:19.20   3rd Qu.:24.00   3rd Qu.:215.0   3rd Qu.:216.0
##  Max.   :25.40   Max.   :54.50   Max.   :233.0   Max.   :240.0
##                                                  NA's   :19
##      Tarsus        WingPitFat      KeelFat          Crop
##  Min.   :24.70   Min.   :0.00   Min.   :0.00   Min.   :0.0000
##  1st Qu.:67.42   1st Qu.:0.00   1st Qu.:2.00   1st Qu.:0.0000
##  Median :70.20   Median :1.00   Median :2.00   Median :0.0000
##  Mean   :65.35   Mean   :0.75   Mean   :2.07   Mean   :0.2372
##  3rd Qu.:73.58   3rd Qu.:1.00   3rd Qu.:3.00   3rd Qu.:0.2500
##  Max.   :75.30   Max.   :2.00   Max.   :3.00   Max.   :1.0000
##  NA's   :62      NA's   :62     NA's   :20     NA's   :21
```

```
hawks_subset <- subset(Hawks, Species %in% c("SS"))
summary(hawks_subset)
```

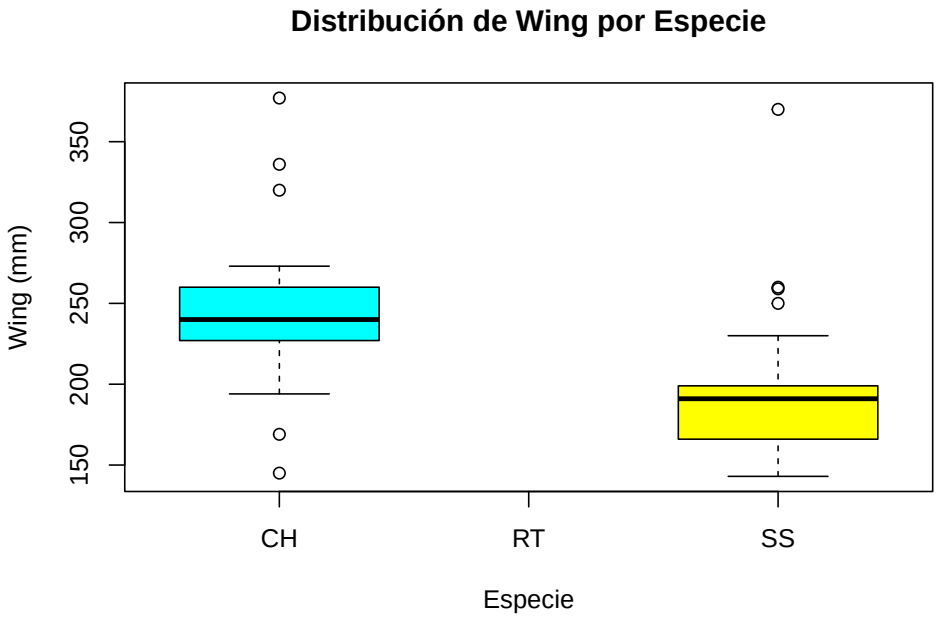
```
##      Month             Day             Year       CaptureTime   ReleaseTime
##  Min.   : 9.000   Min.   : 1.00   Min.   :1992   10:10  :  5          :235
##  1st Qu.: 9.000   1st Qu.:10.00   1st Qu.:1997   10:15  :  5   12:50  :  2
##  Median :10.000   Median :16.00   Median :2000   10:30  :  5   10:20  :  1
##  Mean   : 9.705   Mean   :16.03   Mean   :1999   11:35  :  5   10:32  :  1
##  3rd Qu.:10.000   3rd Qu.:22.00   3rd Qu.:2002   12:10  :  5   10:42  :  1
##  Max.   :11.000   Max.   :31.00   Max.   :2003   12:00  :  4   10:50  :  1
##                                                  (Other):232   (Other): 20
##       BandNumber  Species  Age     Sex          Wing           Weight
##  1142-09240:  1   CH:  0   A: 69    :  1   Min.   :143.0   Min.   :  85.0
##  1142-09241:  1   RT:  0   I:192   F:136   1st Qu.:166.0   1st Qu.: 100.0
##  1142-09242:  1   SS:261           M:124   Median :191.0   Median : 155.0
##  1142-18229:  1                           Mean   :184.9   Mean   : 148.0
##  1142-19209:  1                           3rd Qu.:199.0   3rd Qu.: 177.8
##  1142-19210:  1                           Max.   :370.0   Max.   :1094.0
##  (Other)   :255                                           NA's   :5
```

```
##      Culmen          Hallux            Tail        StandardTail
##  Min.   : 8.60   Min.   :  9.50   Min.   :119.0   Min.   :115.0
##  1st Qu.:10.00   1st Qu.: 11.50   1st Qu.:133.0   1st Qu.:137.0
##  Median :11.70   Median : 13.95   Median :150.0   Median :152.0
##  Mean   :11.48   Mean   : 14.96   Mean   :146.7   Mean   :151.3
##  3rd Qu.:12.30   3rd Qu.: 14.78   3rd Qu.:158.0   3rd Qu.:162.0
##  Max.   :27.30   Max.   :143.00   Max.   :221.0   Max.   :313.0
##  NA's   :3       NA's   :3                        NA's   :68
##      Tarsus        WingPitFat       KeelFat           Crop
##  Min.   :44.50   Min.   :0.000   Min.   :0.000   Min.   :0.0000
##  1st Qu.:50.15   1st Qu.:1.000   1st Qu.:2.000   1st Qu.:0.0000
##  Median :54.45   Median :1.000   Median :3.000   Median :0.0000
##  Mean   :53.34   Mean   :1.345   Mean   :2.763   Mean   :0.3294
##  3rd Qu.:56.50   3rd Qu.:2.000   3rd Qu.:4.000   3rd Qu.:0.5000
##  Max.   :58.10   Max.   :3.000   Max.   :4.000   Max.   :4.0000
##  NA's   :233     NA's   :232     NA's   :67      NA's   :68
```

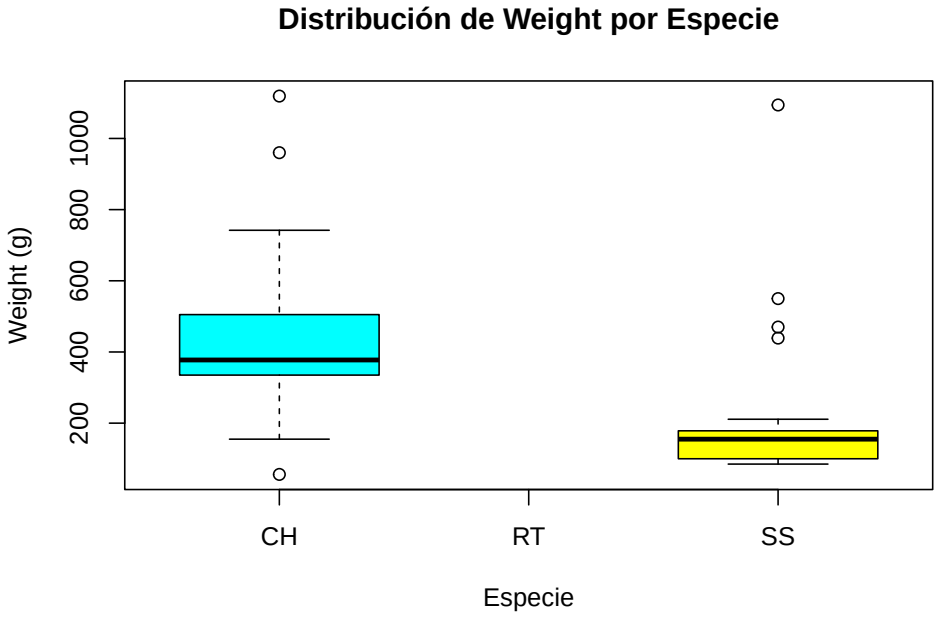
```
hawks_subset <- subset(Hawks, Species %in% c("CH", "SS"))
summary(hawks_subset)
```

```
##      Month            Day             Year       CaptureTime   ReleaseTime
##  Min.   : 9.000   Min.   : 1.00   Min.   :1992   10:10  :  5          :300
##  1st Qu.: 9.000   1st Qu.:10.00   1st Qu.:1997   10:15  :  5   12:50  :  2
##  Median :10.000   Median :16.00   Median :2000   10:30  :  5   14:45  :  2
##  Mean   : 9.689   Mean   :15.96   Mean   :1999   11:35  :  5   10:20  :  1
##  3rd Qu.:10.000   3rd Qu.:23.00   3rd Qu.:2002   12:10  :  5   10:32  :  1
##  Max.   :11.000   Max.   :31.00   Max.   :2003   12:20  :  5   10:42  :  1
##                                                  (Other):301   (Other): 24
##       BandNumber  Species  Age     Sex          Wing            Weight
##             :  1   CH: 70   A:101    :  3   Min.   :143.0   Min.   :  56.0
##  1142-09240:  1   RT:  0   I:230   F:171   1st Qu.:168.2   1st Qu.: 101.2
##  1142-09241:  1   SS:261           M:157   Median :194.5   Median : 168.0
##  1142-09242:  1                           Mean   :197.3   Mean   : 206.5
##  1142-18229:  1                           3rd Qu.:207.8   3rd Qu.: 195.8
##  1142-19209:  1                           Max.   :377.0   Max.   :1119.0
##  (Other)   :325                           NA's   :1       NA's   :5
##      Culmen          Hallux            Tail        StandardTail
##  Min.   : 8.60   Min.   :  9.50   Min.   :119.0   Min.   :115.0
##  1st Qu.:10.20   1st Qu.: 11.70   1st Qu.:135.5   1st Qu.:139.0
##  Median :12.10   Median : 14.40   Median :155.0   Median :159.0
##  Mean   :12.78   Mean   : 16.64   Mean   :158.2   Mean   :162.9
##  3rd Qu.:13.20   3rd Qu.: 15.60   3rd Qu.:164.0   3rd Qu.:170.2
##  Max.   :27.30   Max.   :143.00   Max.   :233.0   Max.   :313.0
##  NA's   :3       NA's   :3                        NA's   :87
##      Tarsus        WingPitFat       KeelFat           Crop
##  Min.   :24.70   Min.   :0.000   Min.   :0.000   Min.   :0.0000
##  1st Qu.:50.42   1st Qu.:0.000   1st Qu.:2.000   1st Qu.:0.0000
##  Median :55.15   Median :1.000   Median :3.000   Median :0.0000
##  Mean   :56.01   Mean   :1.216   Mean   :2.621   Mean   :0.3108
##  3rd Qu.:57.52   3rd Qu.:2.000   3rd Qu.:3.000   3rd Qu.:0.5000
##  Max.   :75.30   Max.   :3.000   Max.   :4.000   Max.   :4.0000
##  NA's   :295     NA's   :294     NA's   :87      NA's   :89
```

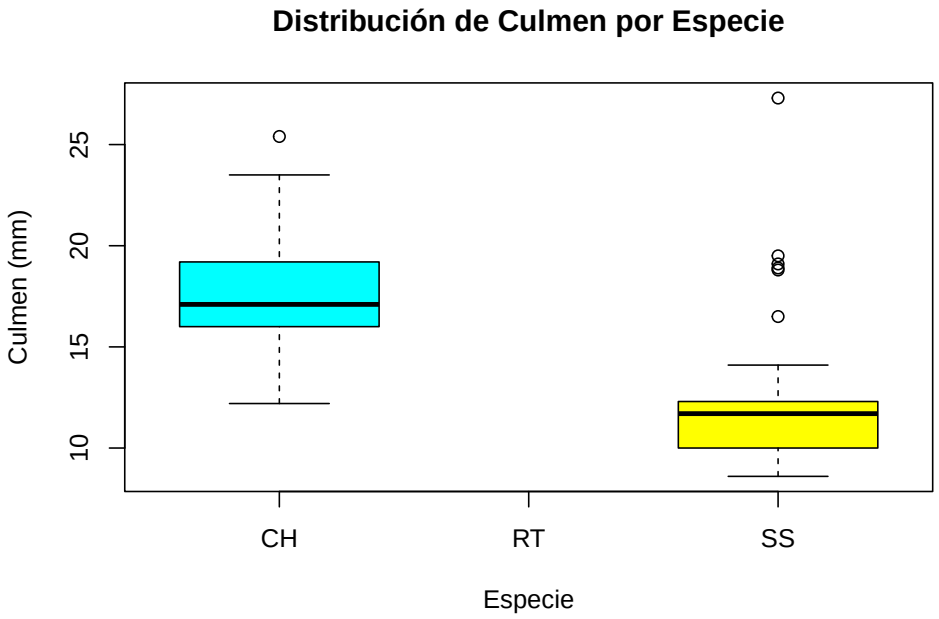
```r
boxplot(Wing ~ Species, data = hawks_subset, main = "Distribución de Wing por Especie",
        ylab = "Wing (mm)", xlab = "Especie", col = c("cyan", "brown", "yellow"))
```



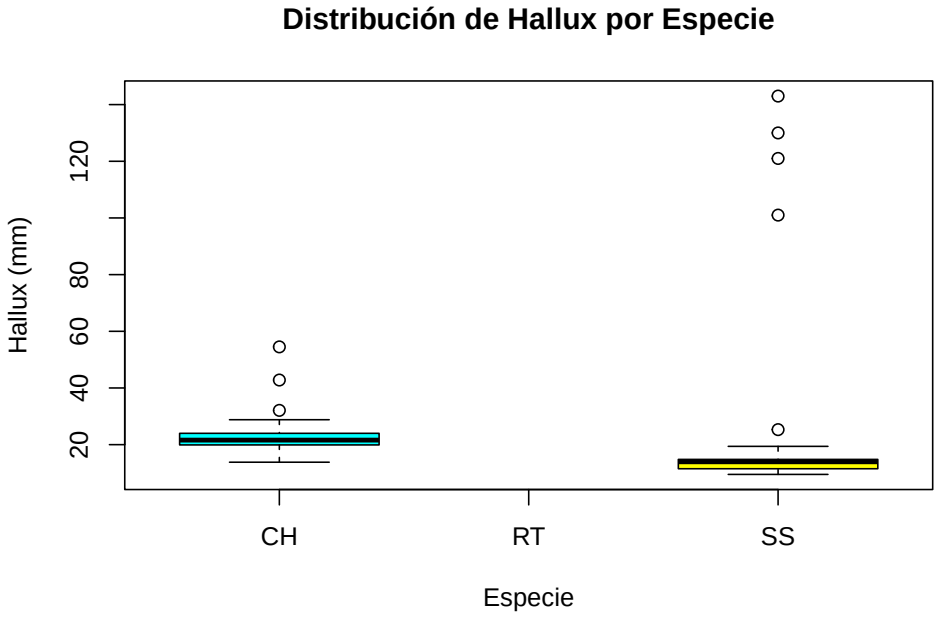

```r
boxplot(Weight ~ Species, data = hawks_subset, main = "Distribución de Weight por Especie",
        ylab = "Weight (g)", xlab = "Especie", col = c("cyan", "brown", "yellow"))
```



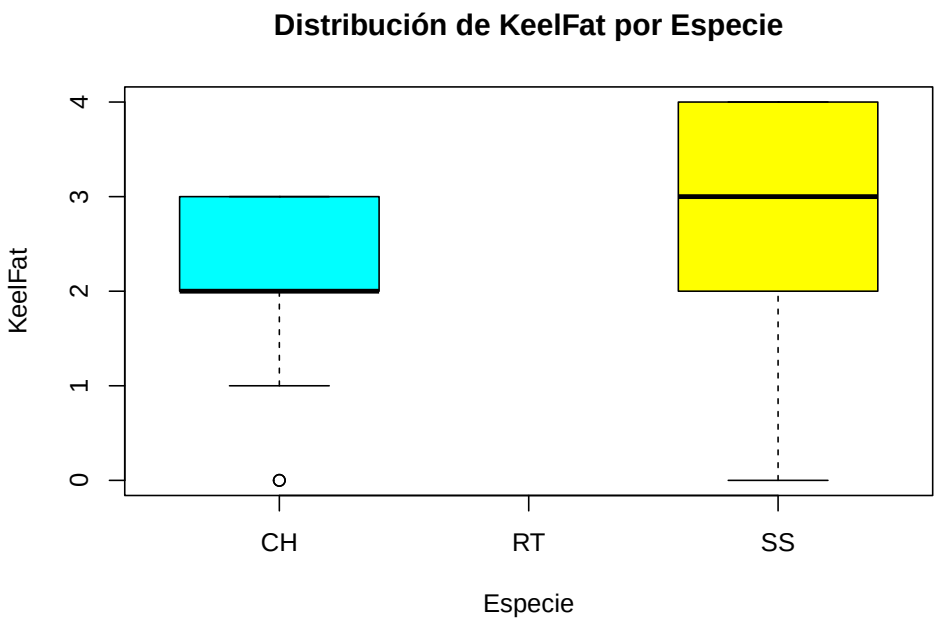
```r
boxplot(Culmen ~ Species, data = hawks_subset, main = "Distribución de Culmen por Especie",
        ylab = "Culmen (mm)", xlab = "Especie", col = c("cyan", "brown", "yellow"))
```



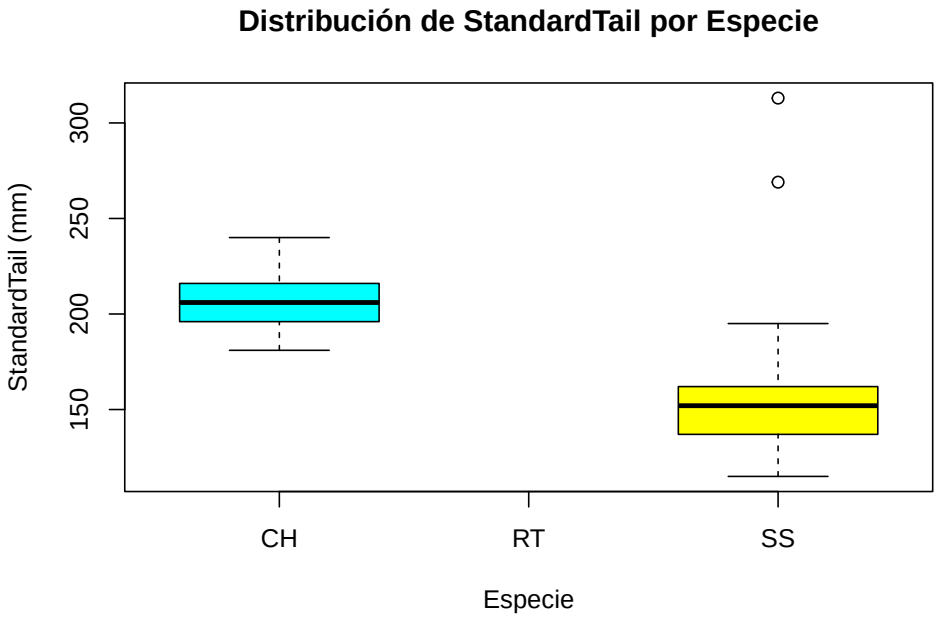

```r
boxplot(Hallux ~ Species, data = hawks_subset, main = "Distribución de Hallux por Especie",
        ylab = "Hallux (mm)", xlab = "Especie", col = c("cyan", "brown", "yellow"))
```



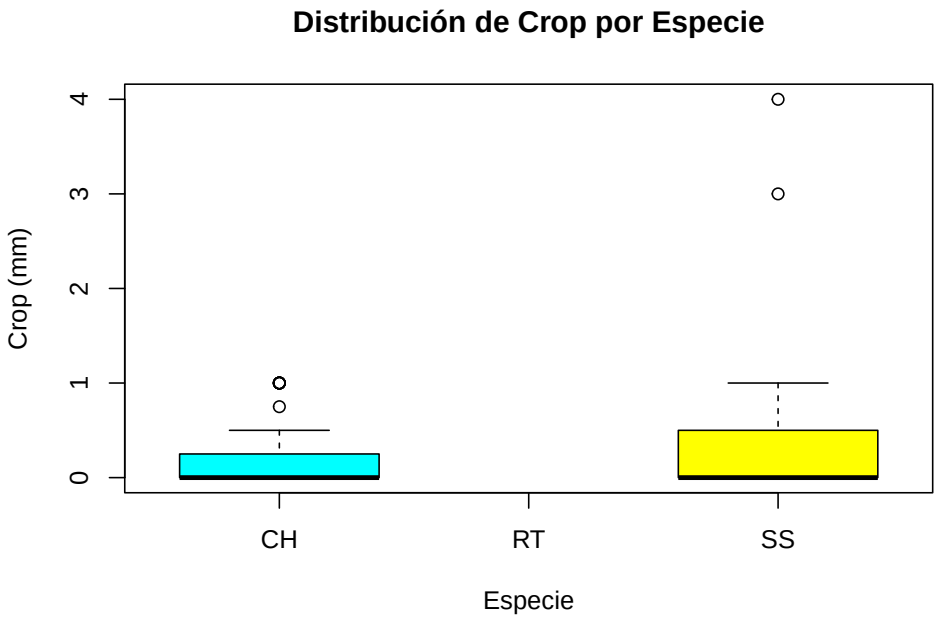
```r
boxplot(KeelFat ~ Species, data = hawks_subset, main = "Distribución de KeelFat por Especie",
        ylab = "KeelFat", xlab = "Especie", col = c("cyan", "brown", "yellow"))
```




```r
boxplot(StandardTail ~ Species, data = hawks_subset, main = "Distribución de StandardTail por Especie",
        ylab = "StandardTail (mm)", xlab = "Especie", col = c("cyan", "brown", "yellow"))
```



```
boxplot(Crop ~ Species, data = hawks_subset, main = "Distribución de Crop por Especie",
        ylab = "Crop (mm)", xlab = "Especie", col = c("cyan", "brown", "yellow"))
```



Aquí podemos observar, que hay una clara distribución y concentración de valores según la escecie que se trate, algo evidente y bastante interesante para análisis posteriores. Ademas si separamos RT de CH y SS, y ponemos la lupa en esas dos, podemos ver como Wing no ayudará suficiente para separar CH de SS, ya que sus bigotes del boxplot terminan en la caja de este. Si bien aquí podríamos llegar a casi separar todos los grupos, el análisis se puede volver muy largo ya que el siguiente paso que haría sería ver si edad afecta a las variables.

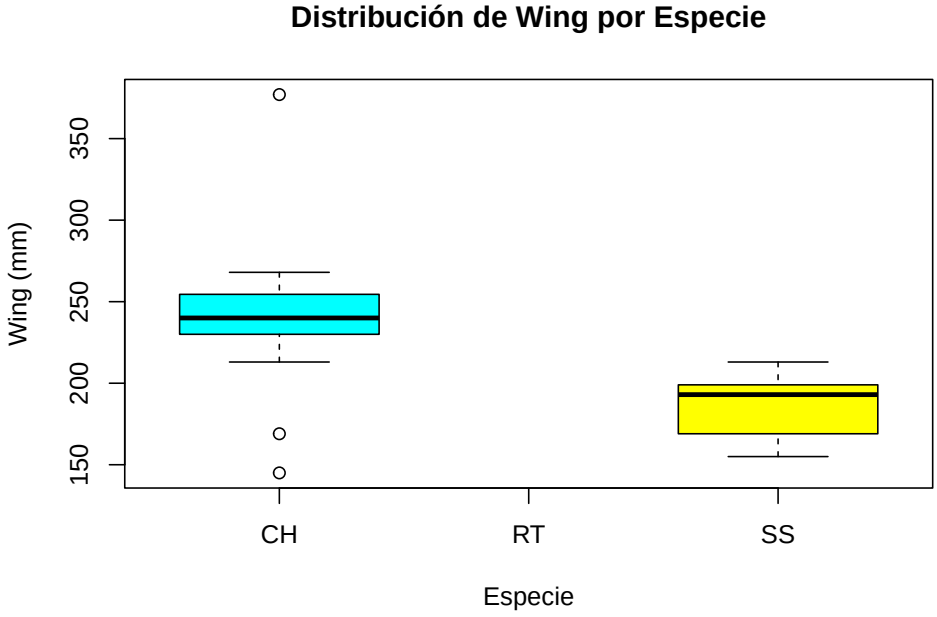
Boxplot de CH y SS filtrando por adultos

```
hawks_subset <- subset(hawks_subset, Age %in% c("A"))
summary(hawks_subset)
```

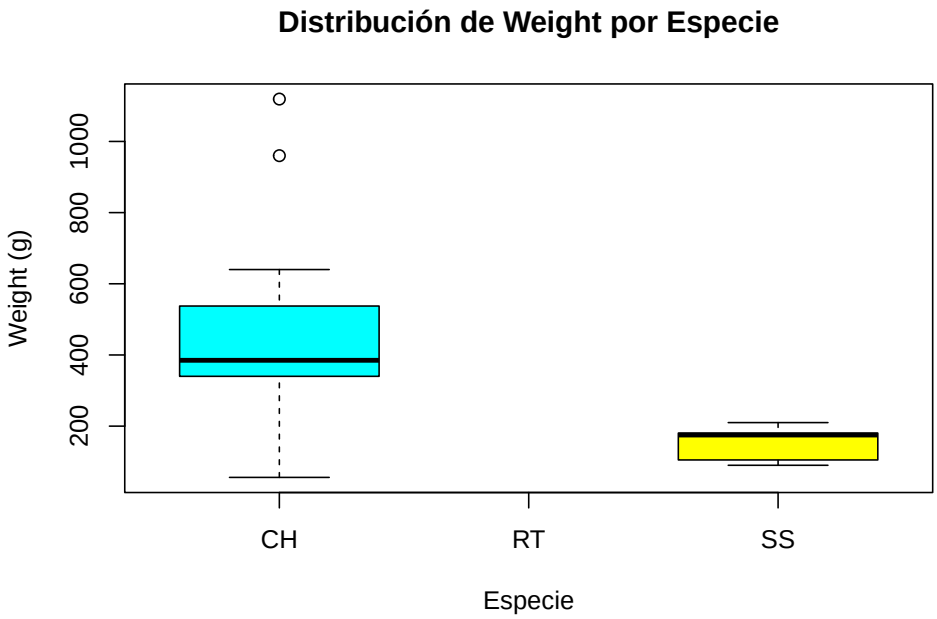
```
##      Month            Day            Year       CaptureTime  ReleaseTime
##  Min.   : 9.000   Min.   : 1.00   Min.   :1993   11:35  : 3          :98
##  1st Qu.:10.000   1st Qu.: 8.00   1st Qu.:1999   12:10  : 3   11:50  : 1
##  Median :10.000   Median :14.00   Median :2000   10:10  : 2   12:30  : 1
##  Mean   : 9.822   Mean   :14.36   Mean   :1999   10:11  : 2   15:00  : 1
##  3rd Qu.:10.000   3rd Qu.:20.00   3rd Qu.:2001   10:45  : 2          : 0
##  Max.   :11.000   Max.   :30.00   Max.   :2003   12:35  : 2   10:20  : 0
##                                                  (Other):87   (Other): 0
##       BandNumber Species Age     Sex         Wing            Weight
##            : 1   CH:32   A:101    : 1   Min.   :145.0   Min.   :  56.0
##  1142-09240: 1   RT: 0   I:  0   F:58   1st Qu.:171.8   1st Qu.: 117.5
##  1142-09241: 1   SS:69           M:42   Median :197.5   Median : 180.0
##  1142-19209: 1                          Mean   :203.3   Mean   : 246.4
##  1142-19214: 1                          3rd Qu.:230.0   3rd Qu.: 339.2
##  1142-19215: 1                          Max.   :377.0   Max.   :1119.0
```

```
##  (Other)   :95                          NA's   :1       NA's   :1
##      Culmen          Hallux            Tail         StandardTail
##  Min.   : 9.30   Min.   : 10.80   Min.   :119.0   Min.   :121.0
##  1st Qu.:10.90   1st Qu.: 13.40   1st Qu.:138.0   1st Qu.:146.0
##  Median :12.70   Median : 14.80   Median :156.0   Median :160.0
##  Mean   :13.69   Mean   : 19.14   Mean   :162.2   Mean   :168.8
##  3rd Qu.:16.10   3rd Qu.: 19.90   3rd Qu.:185.0   3rd Qu.:195.0
##  Max.   :25.40   Max.   :143.00   Max.   :233.0   Max.   :269.0
##                                                   NA's   :20
##      Tarsus        WingPitFat     KeelFat           Crop
##  Min.   :24.70   Min.   :0.0   Min.   :1.000   Min.   :0.0000
##  1st Qu.:47.12   1st Qu.:0.0   1st Qu.:2.000   1st Qu.:0.0000
##  Median :54.80   Median :0.5   Median :3.000   Median :0.0000
##  Mean   :51.38   Mean   :1.0   Mean   :2.833   Mean   :0.3635
##  3rd Qu.:59.05   3rd Qu.:1.5   3rd Qu.:4.000   3rd Qu.:1.0000
##  Max.   :71.20   Max.   :3.0   Max.   :4.000   Max.   :1.0000
##  NA's   :97      NA's   :97    NA's   :20      NA's   :21
```

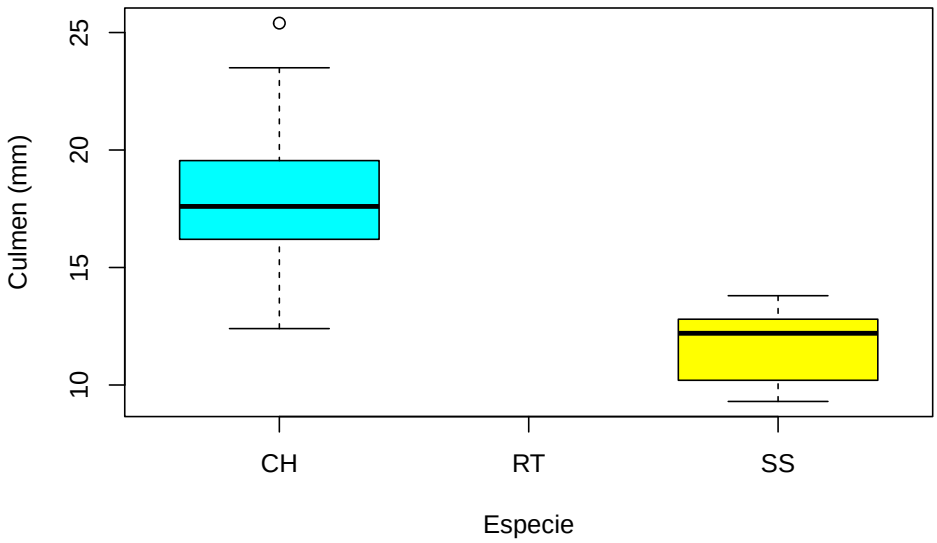
```r
boxplot(Wing ~ Species, data = hawks_subset, main = "Distribución de Wing por Especie",
        ylab = "Wing (mm)", xlab = "Especie", col = c("cyan", "brown", "yellow"))
```



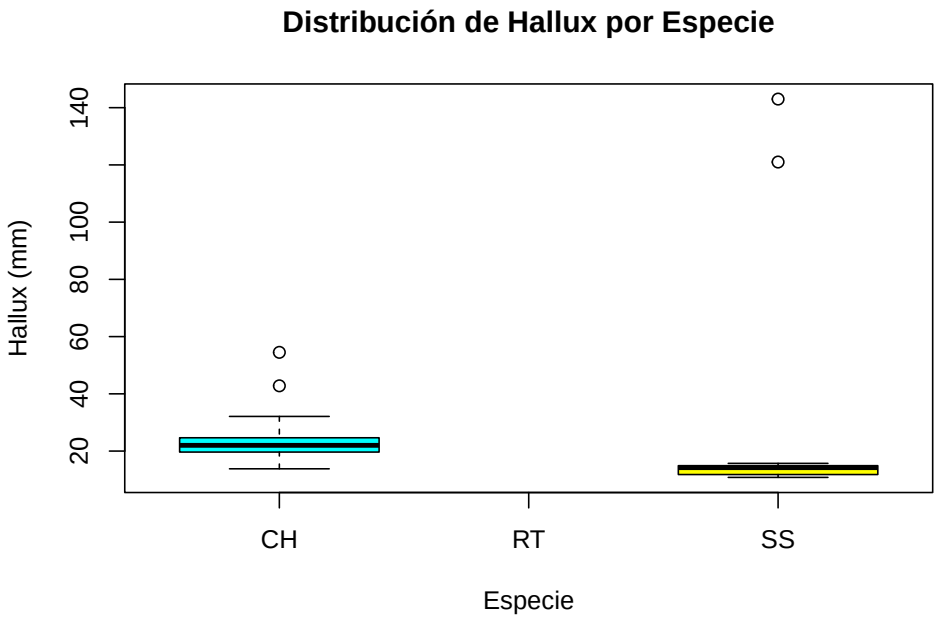

```r
boxplot(Weight ~ Species, data = hawks_subset, main = "Distribución de Weight por Especie",
        ylab = "Weight (g)", xlab = "Especie", col = c("cyan", "brown", "yellow"))
```



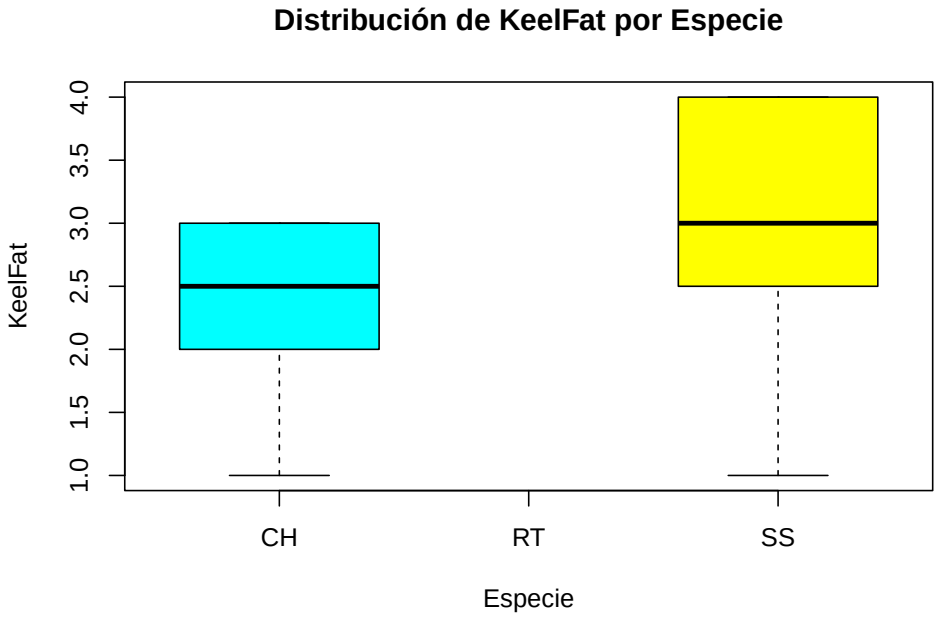

```
boxplot(Culmen ~ Species, data = hawks_subset, main = "Distribución de Culmen por Especie",
        ylab = "Culmen (mm)", xlab = "Especie", col = c("cyan", "brown", "yellow"))
```



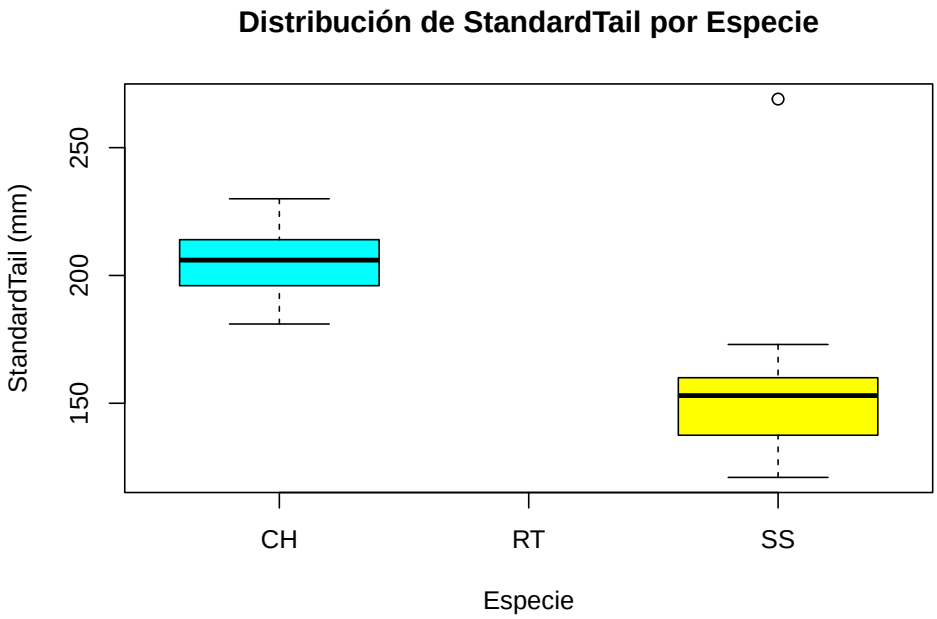

```
boxplot(Hallux ~ Species, data = hawks_subset, main = "Distribución de Hallux por Especie",
        ylab = "Hallux (mm)", xlab = "Especie", col = c("cyan", "brown", "yellow"))
```



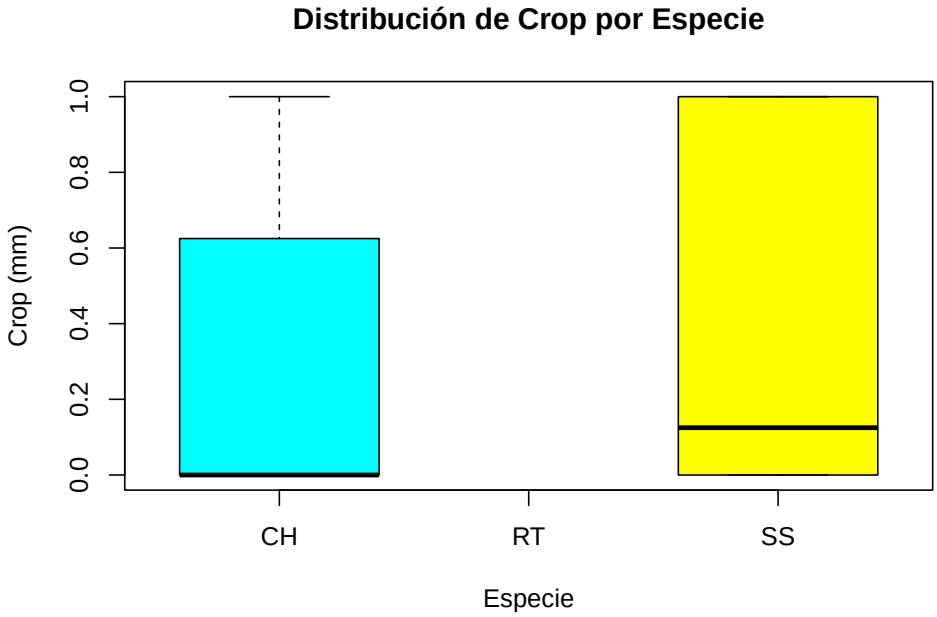

```
boxplot(KeelFat ~ Species, data = hawks_subset, main = "Distribución de KeelFat por Especie",
        ylab = "KeelFat", xlab = "Especie", col = c("cyan", "brown", "yellow"))
```




```
boxplot(StandardTail ~ Species, data = hawks_subset, main = "Distribución de StandardTail por Especie",
        ylab = "StandardTail (mm)", xlab = "Especie", col = c("cyan", "brown", "yellow"))
```



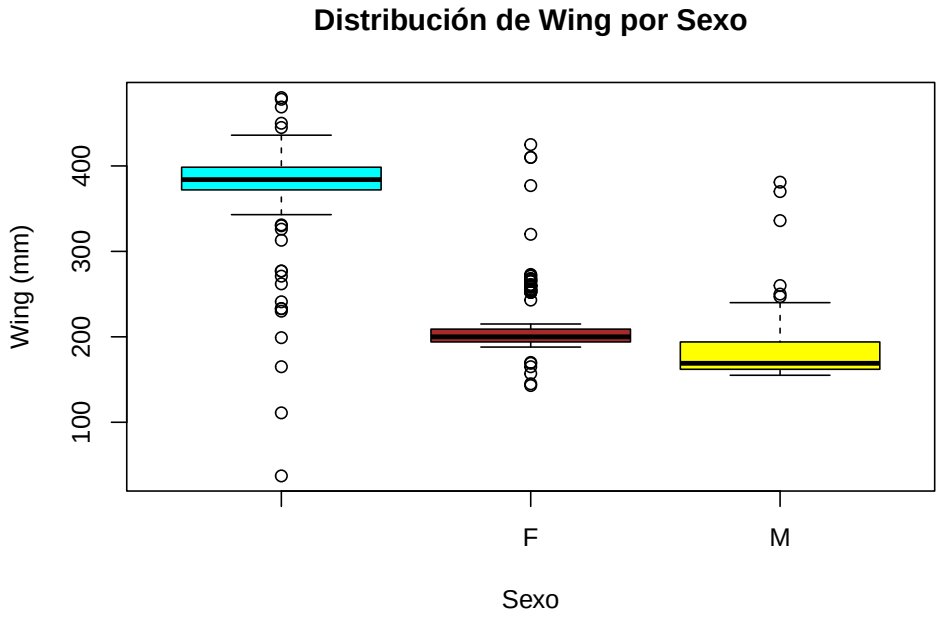

```
boxplot(Crop ~ Species, data = hawks_subset, main = "Distribución de Crop por Especie",
        ylab = "Crop (mm)", xlab = "Especie", col = c("cyan", "brown", "yellow"))
```



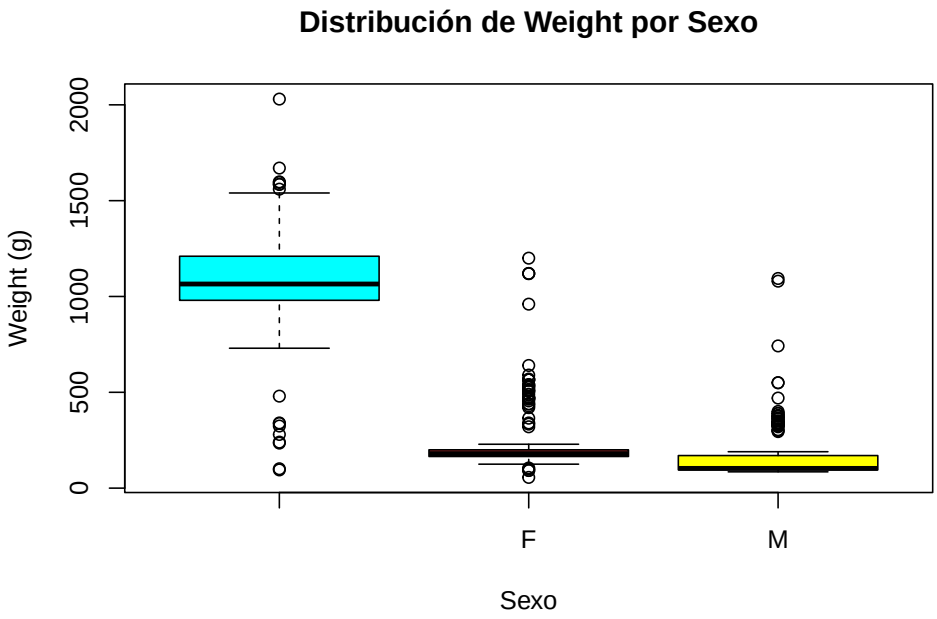
Si bien para este análisis no nos sirve, ya que perderíamos una parte importante del dataset, mas de 200 individuos, es interesante saber que, en los adultos podemos tener identificadas las 3 especies, sea por kmeans u otro método, ya que tenemos bien acotadas las medidas en Wing y StandardTail para diferenciar estas 2 especies, sumado al análisis exploratorio que ya he hecho de RT, puedo llegar a la conclusión que podría hacer un modelo eficiente para adultos que clasifique según medidas.

```
boxplot(Wing ~ Sex, data = Hawks, main = "Distribución de Wing por Sexo",
        ylab = "Wing (mm)", xlab = "Sexo", col = c("cyan", "brown", "yellow"))
```

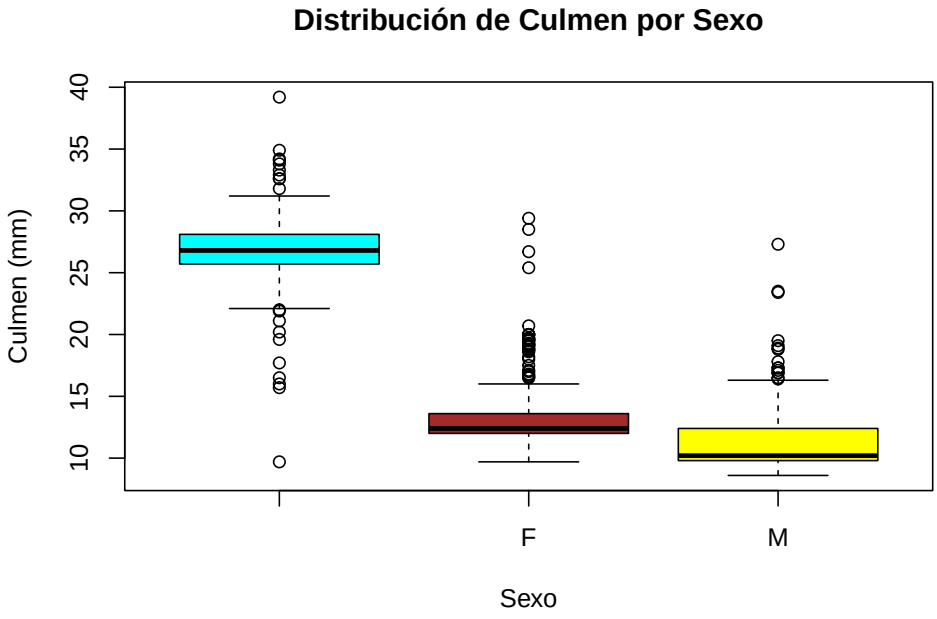
Box plot by Sex.



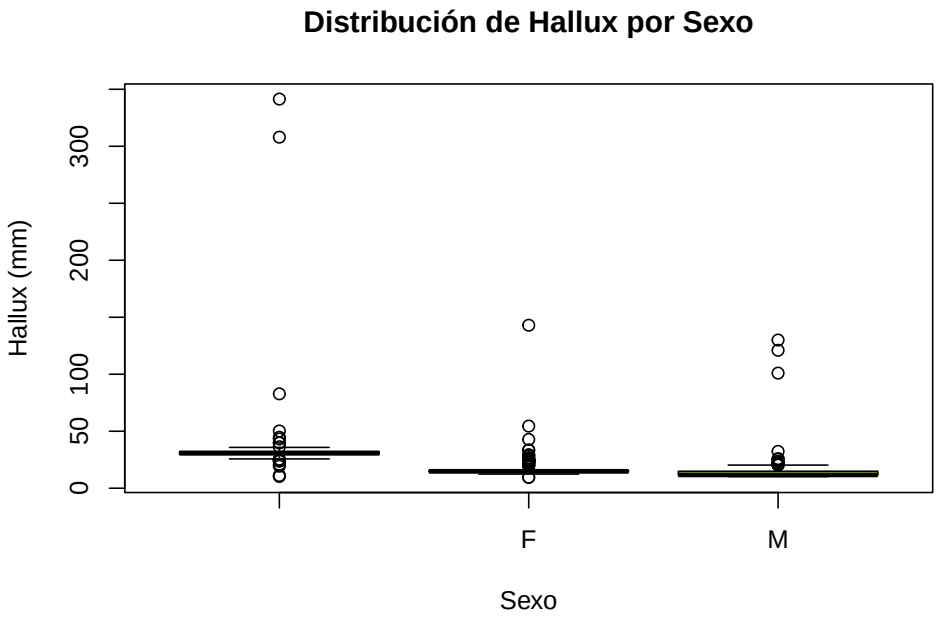

```
boxplot(Weight ~ Sex, data = Hawks, main = "Distribución de Weight por Sexo",
        ylab = "Weight (g)", xlab = "Sexo", col = c("cyan", "brown", "yellow"))
```




```
boxplot(Culmen ~ Sex, data = Hawks, main = "Distribución de Culmen por Sexo",
        ylab = "Culmen (mm)", xlab = "Sexo", col = c("cyan", "brown", "yellow"))
```



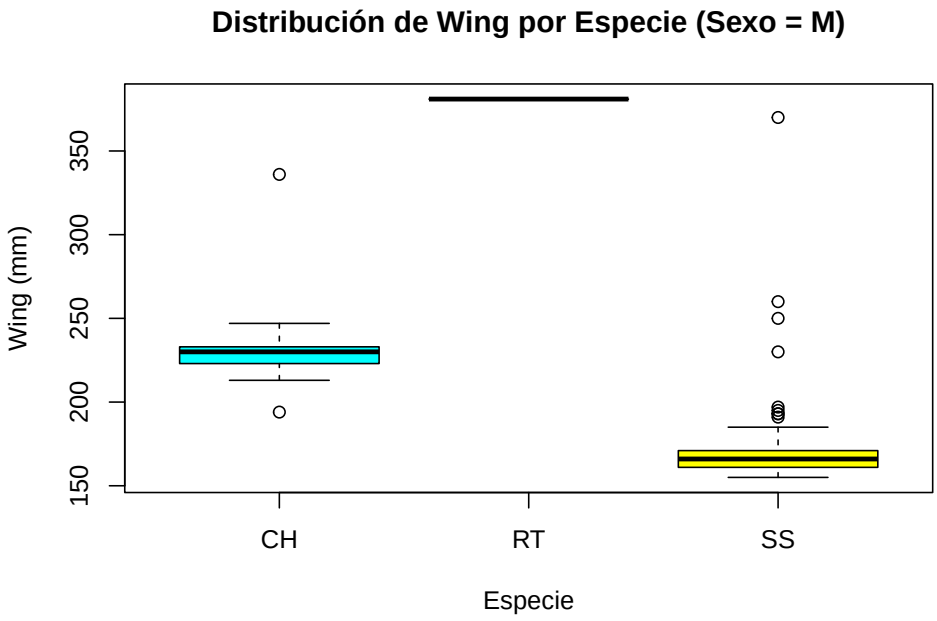

```
boxplot(Hallux ~ Sex, data = Hawks, main = "Distribución de Hallux por Sexo",
        ylab = "Hallux (mm)", xlab = "Sexo", col = c("cyan", "brown", "yellow"))
```



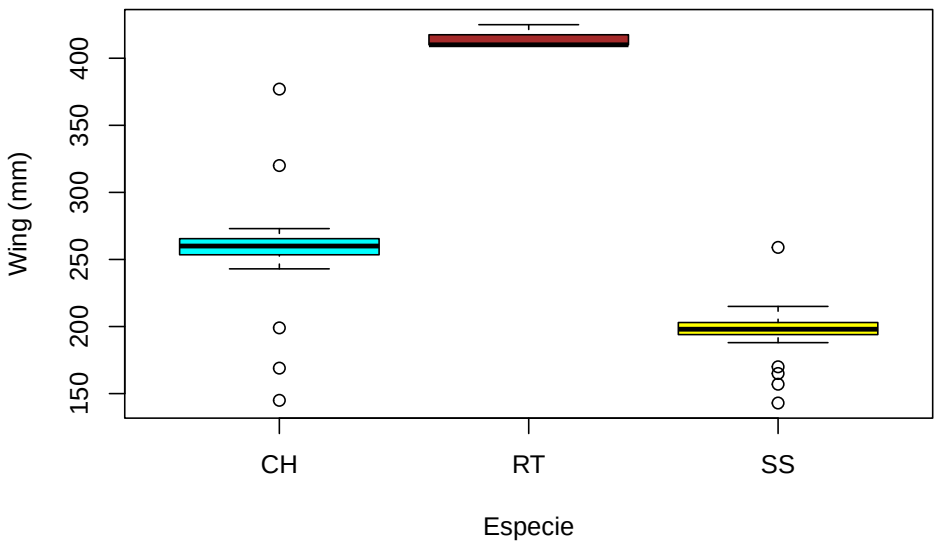
Sin embago si es por sexo, la distribución no está clara. ¿Y si en ciertas especies es más complicado saber el sexo? No tiene sentido que los individuos con sexo conocido se sitúen en una zona concreta y los que no tienen sexo conocido, en otra totalmente diferente.

```
boxplot(Wing ~ Species, data = subset(Hawks, Sex == "M"),
        main = "Distribución de Wing por Especie (Sexo = M)",
        ylab = "Wing (mm)", xlab = "Especie", col = c("cyan", "brown", "yellow"))
```

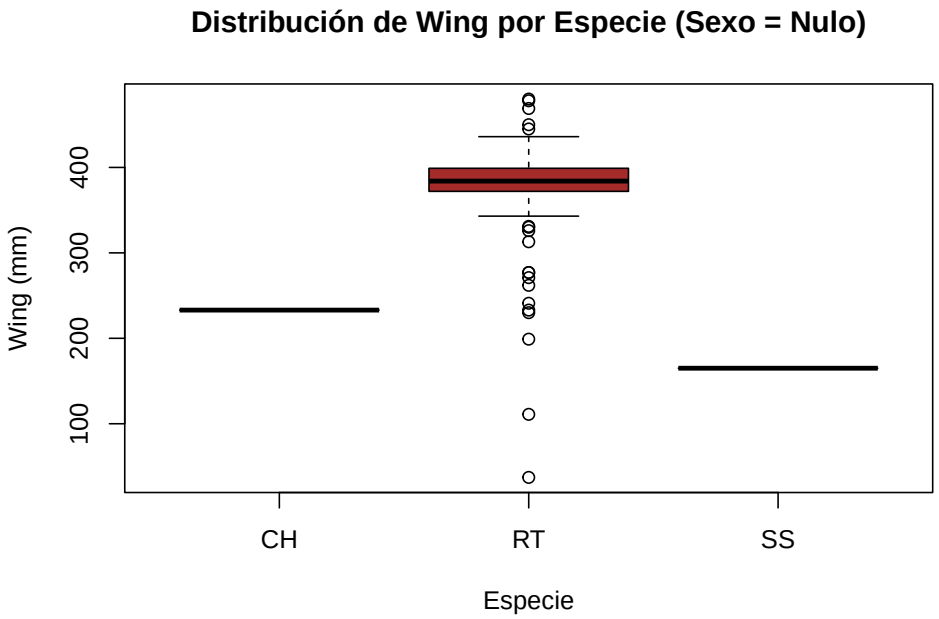
Boxplot by Especie y Sex en Wing




```
boxplot(Wing ~ Species, data = subset(Hawks, Sex == "F"),
        main = "Distribución de Wing por Especie (Sexo = F)",
        ylab = "Wing (mm)", xlab = "Especie", col = c("cyan", "brown", "yellow"))
```




```
boxplot(Wing ~ Species, data = subset(Hawks, Sex == ""),
        main = "Distribución de Wing por Especie (Sexo = Nulo)",
        ylab = "Wing (mm)", xlab = "Especie", col = c("cyan", "brown", "yellow"))
```




```
nulos_sex=subset(Hawks, Sex == "")
conteo_sex <- table(nulos_sex$Sex)
conteo_sex
```

```
##
##       F   M
## 576   0   0
```

```
conteo_especies <- table(nulos_sex$Species)
conteo_especies
```

```
##
##  CH  RT  SS
##   2 573   1
```

```
species_solo_f <- subset(Hawks, Sex == "F" )
species_solo_f <- table(species_solo_f$Species)
species_solo_f
```

```
##
##  CH  RT  SS
##  35   3 136
```

```
species_solo_m <- subset(Hawks, Sex == "M" )
species_solo_m <- table(species_solo_m$Species)
species_solo_m
```

```
##
##  CH  RT  SS
##  33   1 124
```

Bueno, aquí parece que tenemos dos conclusiones, en CH y SS tiene una distribución normal de Sex, sin embargo, en RT, no tenemos Sex, por tanto no podemos usar la especie RT si conlleva utilizar Sex, ya que ni siquiera podríamos hacer imputación de valores tomando valores cercanos ni nada por el estilo ya que no hay apenas muestras en la especie RT...

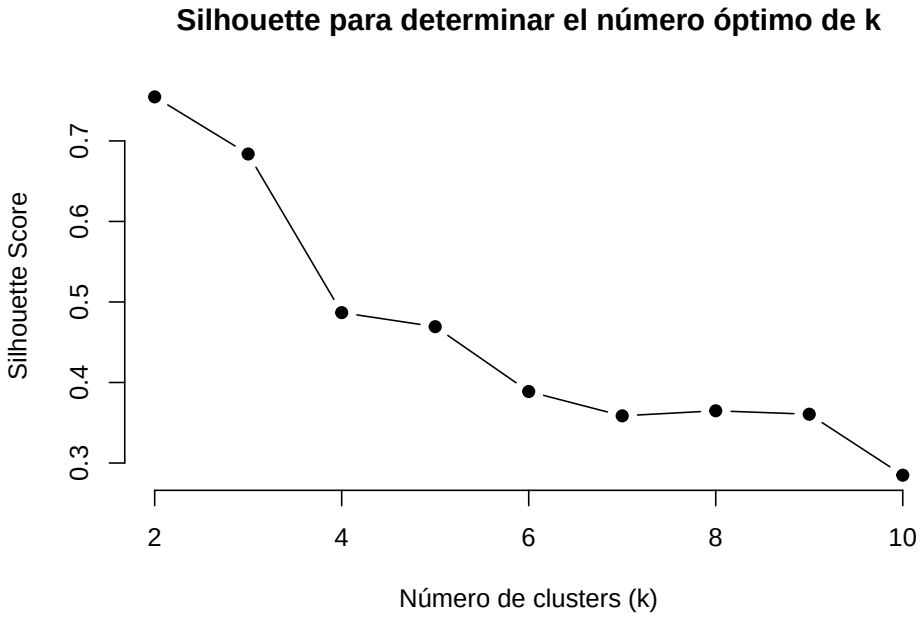
Métodos de búsqueda de valores óptimos de clusters

```
hawks_limpio <- Hawks[, c("Wing", "Weight", "Culmen", "Tail")]#, "Hallux")]
hawks_limpio <- na.omit(hawks_limpio)
hawks_scaled <- scale(hawks_limpio)
```

Limpieza de nulos y escalado

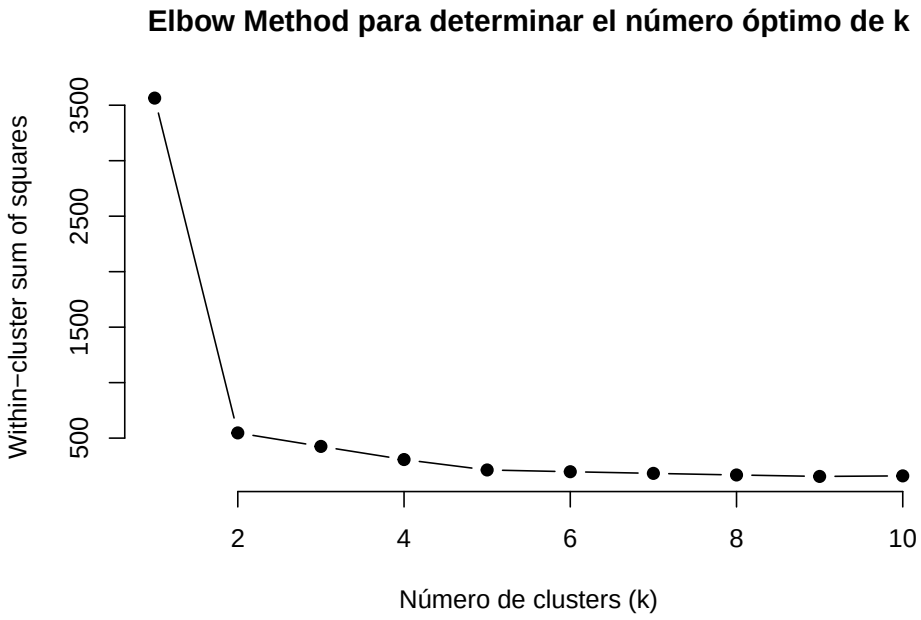
```
max_k <- 10
silhouettes <- numeric(max_k)
for (k in 2:max_k) {
  y_cluster <- kmeans(hawks_scaled, centers = k)$cluster
  silhouettes[k] <- calcular_silueta(y_cluster, hawks_scaled)
}
plot(2:max_k, silhouettes[2:max_k], type = "b", pch = 19, frame = FALSE,
     xlab = "Número de clusters (k)", ylab = "Silhouette Score",
     main = "Silhouette para determinar el número óptimo de k")
```

Métodos de cálculo de valor óptimo de número de clústers



```r
inercia_intracluster <- function(vdata, max_k) {
  wss <- numeric(max_k)
  for (k in 1:max_k) {
    wss[k] <- sum(kmeans(vdata, centers = k)$withinss)
  }
  return(wss)
}
wss <- inercia_intracluster(hawks_scaled, max_k)
```

```r
plot(1:max_k, wss, type = "b", pch = 19, frame = FALSE,
     xlab = "Número de clusters (k)", ylab = "Within-cluster sum of squares",
     main = "Elbow Method para determinar el número óptimo de k")
```

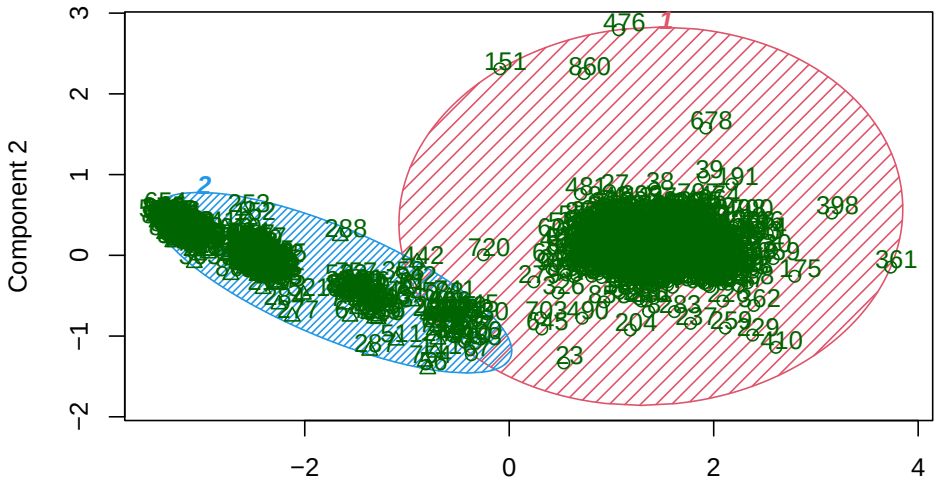


Vemos que en ambos es 2... y en el primero también 3...

```r
usar_kmeans <- function(k, vdata) {
  fit <- kmeans(vdata, centers = k)
  return(fit$cluster)
}
clusters_02 <- usar_kmeans(2,hawks_scaled)
clusters_03 <- usar_kmeans(3,hawks_scaled)

clusplot(hawks_scaled, clusters_02, color=TRUE, shade=TRUE, labels=2, lines=0,
         main="Clusplot: 2 Clusters")
```

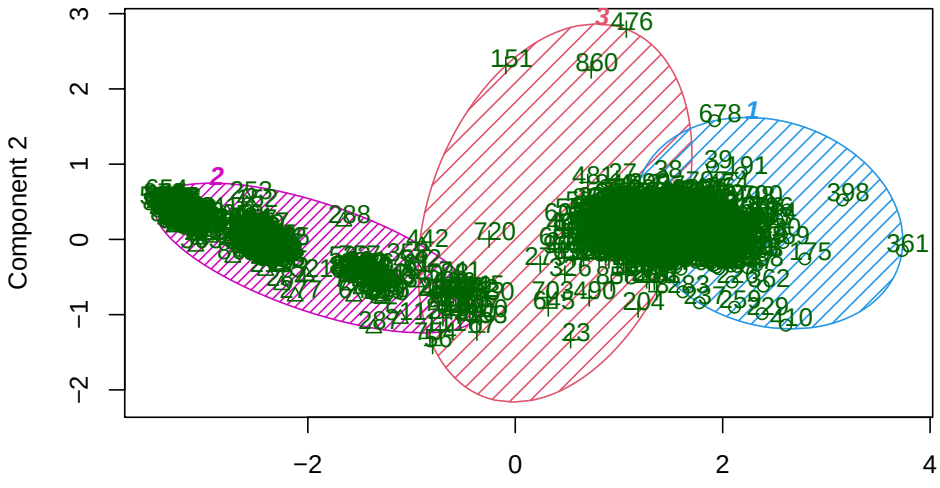
Visualización de 2 y 3 clusters

Clusplot: 2 Clusters




```
clusplot(hawks_scaled, clusters_03, color=TRUE, shade=TRUE, labels=2, lines=0,
         main="Clusplot: 3 Clusters")
```

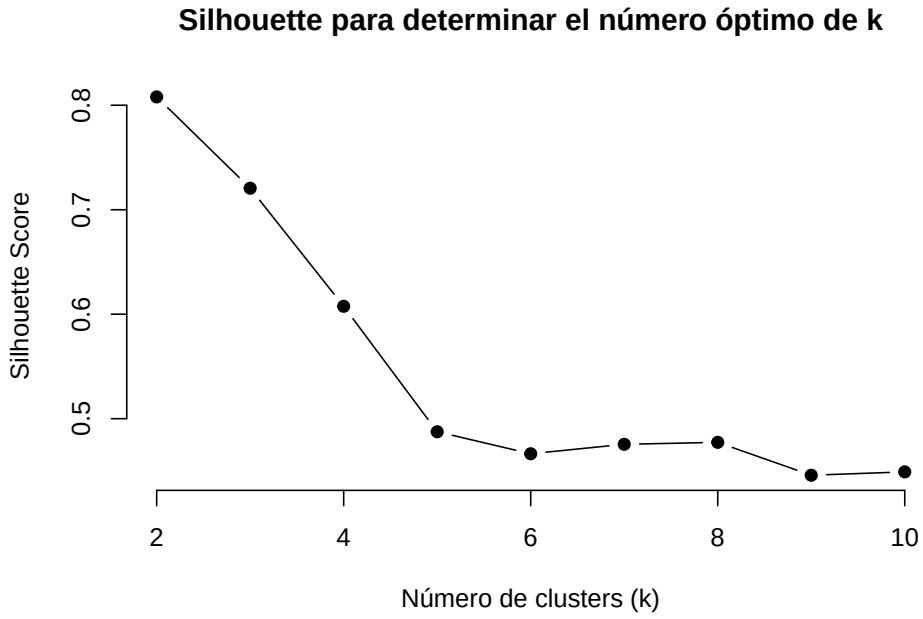
Clusplot: 3 Clusters




```
Hawks$Cluster_A_2 <- NA
Hawks$Cluster_A_3 <- NA
Hawks[rownames(hawks_scaled), "Cluster_A_2"] <- clusters_02
Hawks[rownames(hawks_scaled), "Cluster_A_3"] <- clusters_03
```

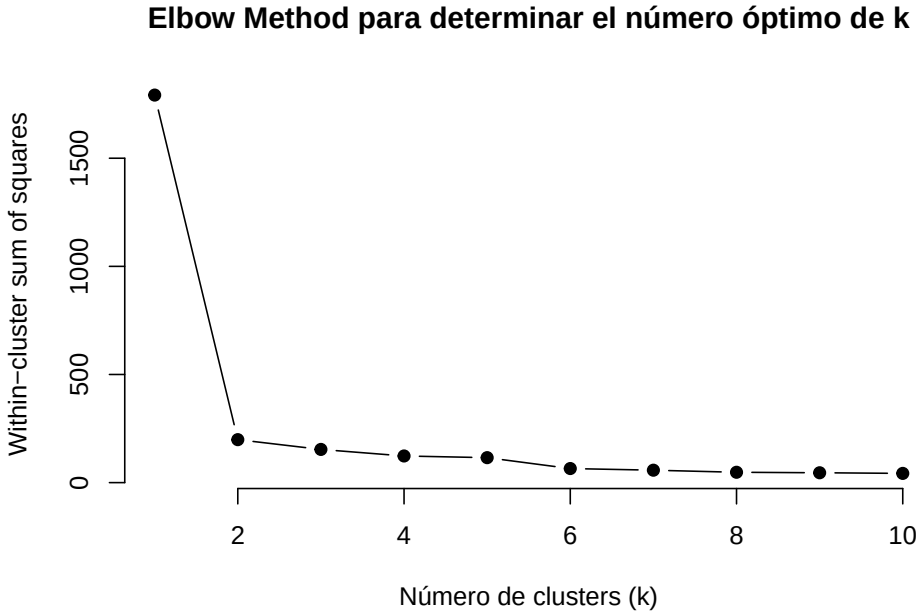
```r
hawks_scaled_menos_variables <-Hawks[ c("Wing", "Weight")]#, "Culmen", "Tail")]#, "Hallux")]
hawks_scaled_menos_variables <- na.omit(hawks_scaled_menos_variables)
hawks_scaled_menos_variables <- scale(hawks_scaled_menos_variables)

max_k <- 10
silhouettes <- numeric(max_k)
for (k in 2:max_k) {
  y_cluster <- kmeans(hawks_scaled_menos_variables, centers = k)$cluster
  silhouettes[k] <- calcular_silueta(y_cluster, hawks_scaled_menos_variables)
}
plot(2:max_k, silhouettes[2:max_k], type = "b", pch = 19, frame = FALSE,
     xlab = "Número de clusters (k)", ylab = "Silhouette Score",
     main = "Silhouette para determinar el número óptimo de k")
```

Vista rápida con menos variables (2 en vez de 5)



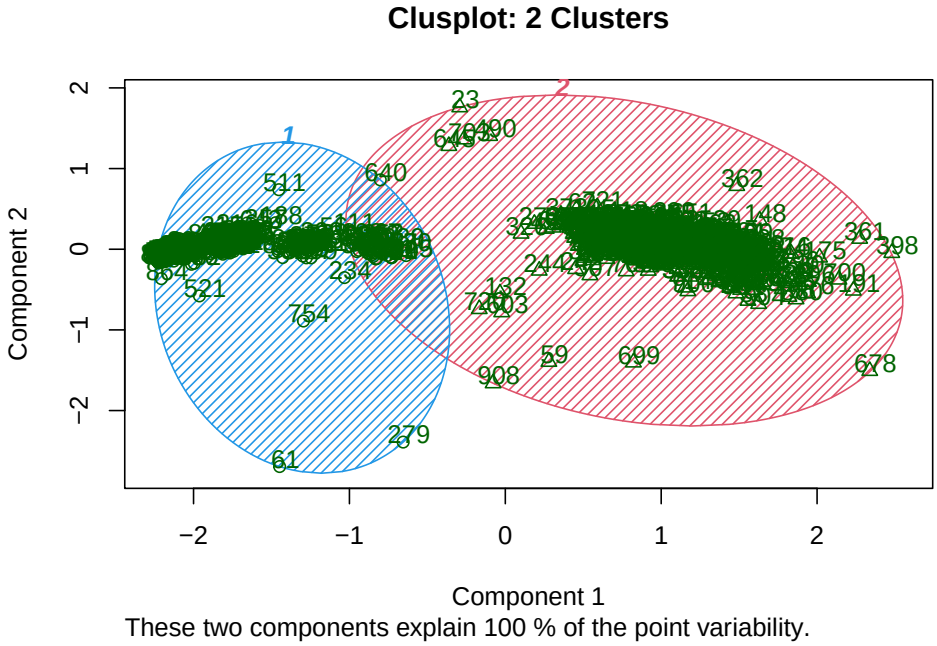

```r
inercia_intracluster <- function(vdata, max_k) {
  wss <- numeric(max_k)
  for (k in 1:max_k) {
    wss[k] <- sum(kmeans(vdata, centers = k)$withinss)
  }
  return(wss)
}
wss <- inercia_intracluster(hawks_scaled_menos_variables, max_k)
```

```r
plot(1:max_k, wss, type = "b", pch = 19, frame = FALSE,
     xlab = "Número de clusters (k)", ylab = "Within-cluster sum of squares",
     main = "Elbow Method para determinar el número óptimo de k")
```

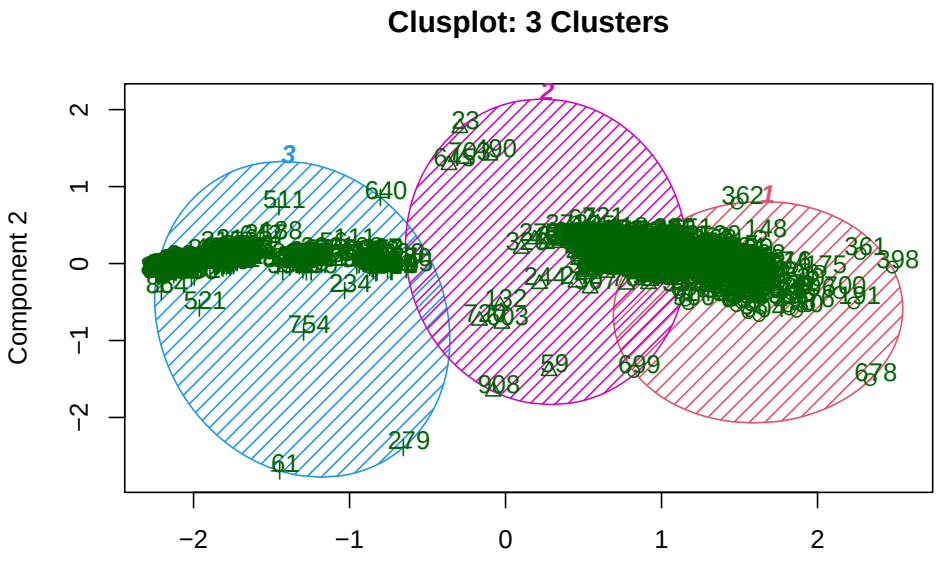



```
clusters_2 <- usar_kmeans(2, hawks_scaled_menos_variables)
clusters_3 <- usar_kmeans(3, hawks_scaled_menos_variables)
```

```
clusplot(hawks_scaled_menos_variables, clusters_2, color=TRUE,
         shade=TRUE, labels=2, lines=0,
         main="Clusplot: 2 Clusters")
```



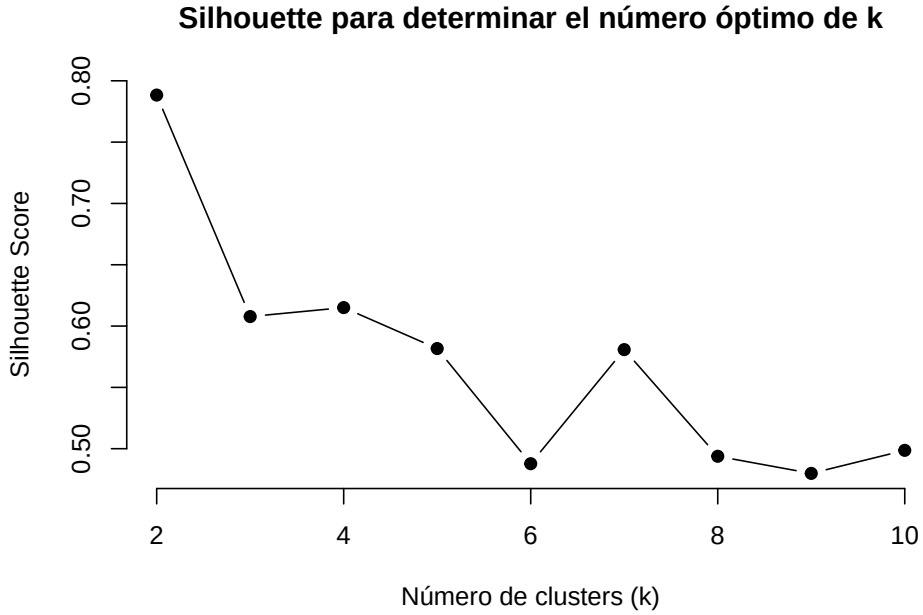
```
clusplot(hawks_scaled_menos_variables, clusters_3, color=TRUE,
         shade=TRUE, labels=2, lines=0,
         main="Clusplot: 3 Clusters")
```



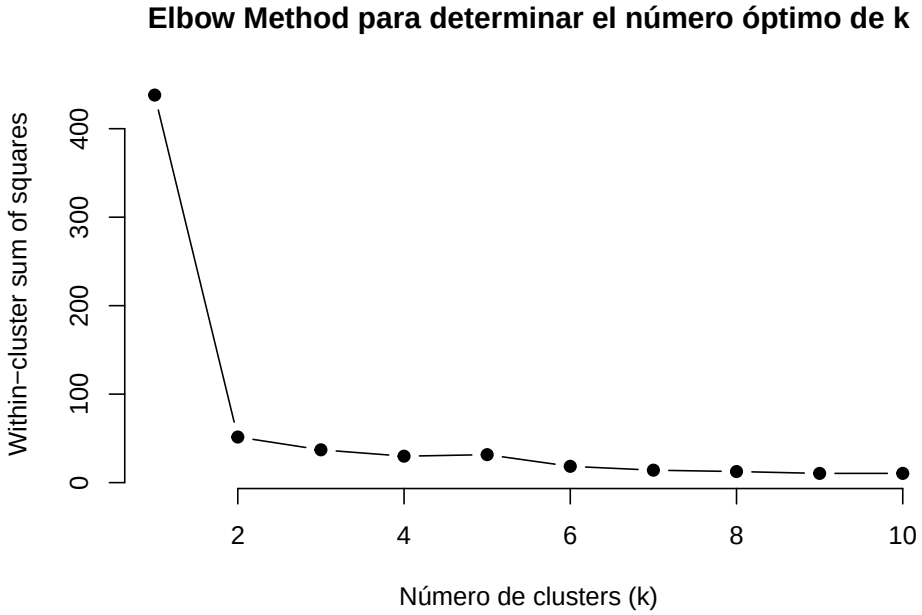

```
Hawks$Cluster_B_2 <- NA
Hawks$Cluster_B_3 <- NA
Hawks[rownames(hawks_scaled_menos_variables), "Cluster_B_2"] <- clusters_2
Hawks[rownames(hawks_scaled_menos_variables), "Cluster_B_3"] <- clusters_3
```

```
Hawks_adultos<-subset(Hawks, Age %in% c("A"))

hawks_scaled_menos_variables <-Hawks_adultos[ c("Weight", "Wing") ]
hawks_scaled_menos_variables <- na.omit(hawks_scaled_menos_variables)
hawks_scaled_menos_variables <- scale(hawks_scaled_menos_variables)

max_k <- 10
silhouettes <- numeric(max_k)
for (k in 2:max_k) {
  y_cluster <- kmeans(hawks_scaled_menos_variables, centers = k)$cluster
  silhouettes[k] <- calcular_silueta(y_cluster, hawks_scaled_menos_variables)
}
plot(2:max_k, silhouettes[2:max_k], type = "b", pch = 19, frame = FALSE,
     xlab = "Número de clusters (k)", ylab = "Silhouette Score",
     main = "Silhouette para determinar el número óptimo de k")
```

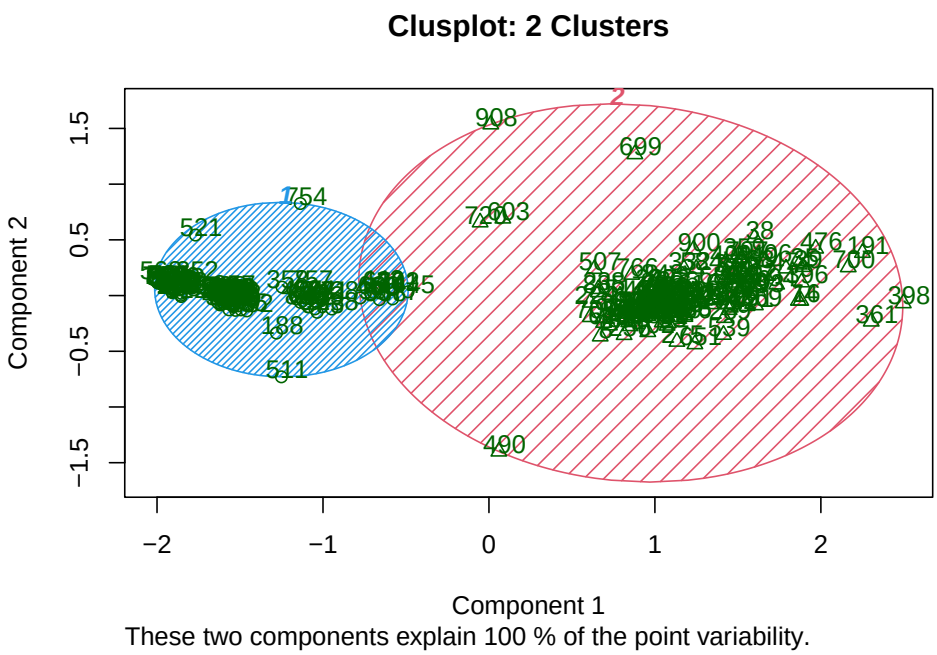
Vista rápida con menos variables(2 en vez de 5) y solo adultos



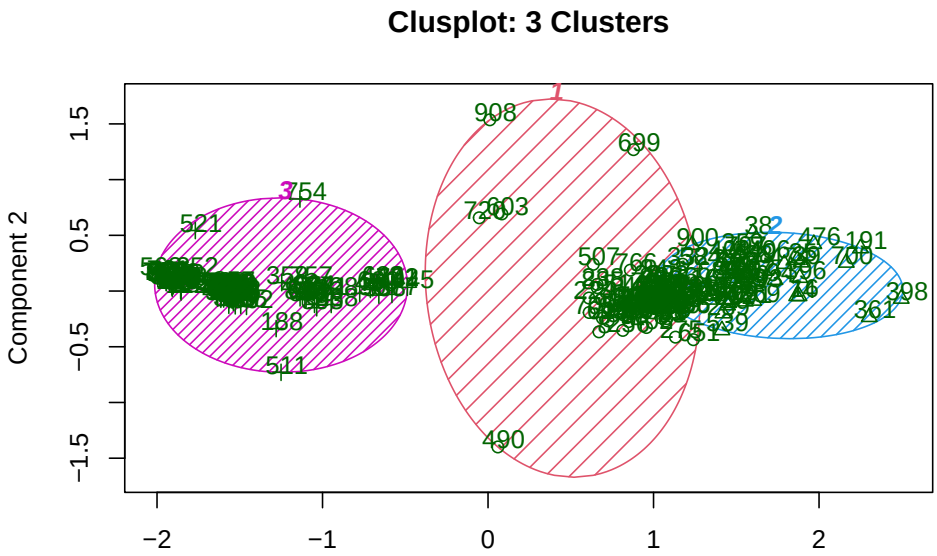

```r
inercia_intracluster <- function(vdata, max_k) {
  wss <- numeric(max_k)
  for (k in 1:max_k) {
    wss[k] <- sum(kmeans(vdata, centers = k)$withinss)
  }
  return(wss)
}
wss <- inercia_intracluster(hawks_scaled_menos_variables, max_k)
plot(1:max_k, wss, type = "b", pch = 19, frame = FALSE,
     xlab = "Número de clusters (k)", ylab = "Within-cluster sum of squares",
     main = "Elbow Method para determinar el número óptimo de k")
```




```
clusters_2 <- usar_kmeans(2, hawks_scaled_menos_variables)
clusters_3 <- usar_kmeans(3, hawks_scaled_menos_variables)
clusplot(hawks_scaled_menos_variables, clusters_2, color=TRUE,
         shade=TRUE, labels=2, lines=0,
         main="Clusplot: 2 Clusters")
```



```
clusplot(hawks_scaled_menos_variables, clusters_3, color=TRUE,
         shade=TRUE, labels=2, lines=0,
         main="Clusplot: 3 Clusters")
```




```
Hawks$Cluster_C_2 <- NA
Hawks$Cluster_C_3 <- NA
Hawks[rownames(hawks_scaled_menos_variables), "Cluster_C_2"] <- clusters_2
Hawks[rownames(hawks_scaled_menos_variables), "Cluster_C_3"] <- clusters_3
```

Podemos observar que hay una diferencia significativa y que podríamos hacer un modelo de clasificación de adultos para las 3 especies casi con seguridad.

Usar kmeansruns

Podemos usar kmeans runs para buscar el numero optimo de clusters, si bien el uso es normalmente antes de clusterizar, lo he querido dejar plasmado para comparar con como lo he hecho, usando los dos criterios utilizados en el ejemplo, silueta media y Calinski-Harabasz, aunque la conclusión hubiera sido parecida...

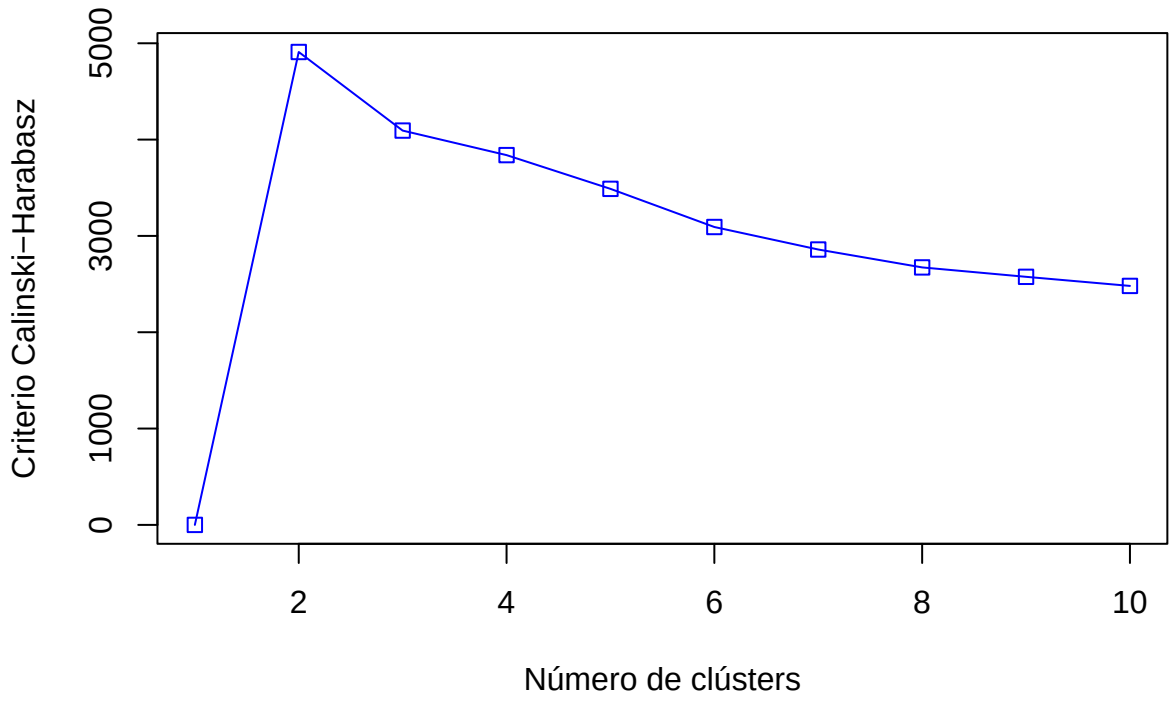
```
fit_ch  <- kmeansruns(hawks_scaled, krange = 1:10, criterion = "ch")
fit_asw <- kmeansruns(hawks_scaled, krange = 1:10, criterion = "asw")
fit_ch$bestk
```

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

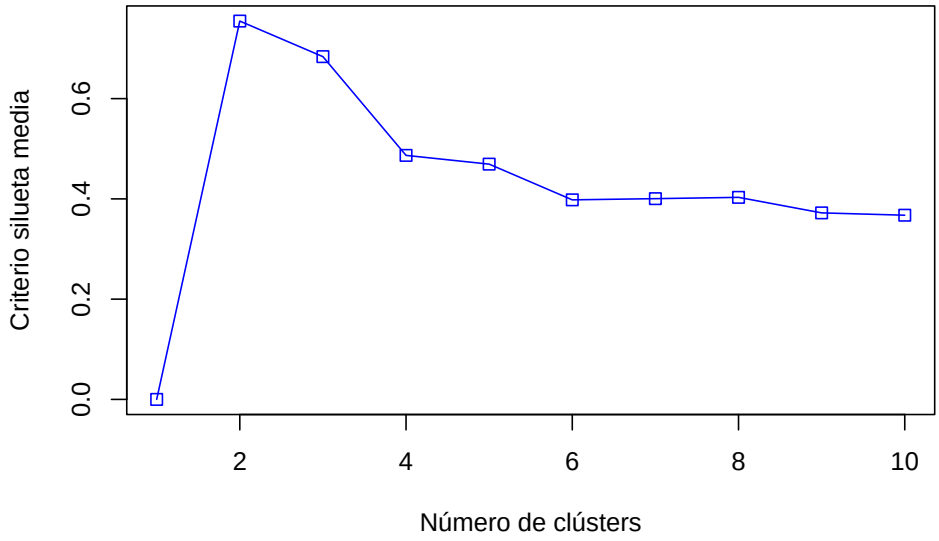
```
fit_asw$bestk
```

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

```
plot(1:10,fit_ch$crit,type="o",col="blue",pch=0,xlab="Número de clústers",
     ylab="Criterio Calinski-Harabasz")
```




```
plot(1:10,fit_asw$crit,type="o",col="blue",pch=0,xlab="Número de clústers",
     ylab="Criterio silueta media")
```



Ver el dataset original

```
head(Hawks)
```

```
##   Month Day Year CaptureTime ReleaseTime  BandNumber Species Age Sex Wing
## 1     9  19 1992       13:30                877-76317      RT   I      385
## 2     9  22 1992       10:30                877-76318      RT   I      376
## 3     9  23 1992       12:45                877-76319      RT   I      381
## 4     9  23 1992       10:50                745-49508      CH   I   F  265
## 5     9  27 1992       11:15               1253-98801      SS   I   F  205
## 6     9  28 1992       11:25              1207-55910       RT   I      412
##   Weight Culmen Hallux Tail StandardTail Tarsus WingPitFat KeelFat Crop
## 1    920   25.7   30.1  219           NA     NA         NA      NA   NA
## 2    930     NA     NA  221           NA     NA         NA      NA   NA
## 3    990   26.7   31.3  235           NA     NA         NA      NA   NA
## 4    470   18.7   23.5  220           NA     NA         NA      NA   NA
## 5    170   12.5   14.3  157           NA     NA         NA      NA   NA
## 6   1090   28.5   32.2  230           NA     NA         NA      NA   NA
##   Cluster_A_2 Cluster_A_3 Cluster_B_2 Cluster_B_3 Cluster_C_2 Cluster_C_3
## 1           1           3           2           2          NA          NA
## 2          NA          NA           2           2          NA          NA
## 3           1           3           2           2          NA          NA
## 4           1           3           1           3          NA          NA
## 5           2           2           1           3          NA          NA
## 6           1           1           2           1          NA          NA
```

Caracteristicas de cada cluster

```
# Datos agregados ponderados de cada cluster
aggregate(. ~ Cluster_A_2, data = Hawks[, c("Wing",
```

```
                                          "Weight",
                                          "Culmen",
                                          "Tail",
                                          "Hallux",
                                          "Cluster_A_2")], mean)
```

```
##   Cluster_A_2     Wing    Weight   Culmen     Tail   Hallux
## 1           1 381.5309 1082.6031 26.82165 222.1478 31.75576
## 2           2 192.4214  185.8706 12.36748 155.0939 16.35065
```

```
aggregate(. ~ Cluster_A_3, data = Hawks[, c("Wing",
                                          "Weight",
                                          "Culmen",
                                          "Tail",
                                          "Hallux",
                                          "Cluster_A_3")], mean)
```

```
##   Cluster_A_3     Wing    Weight   Culmen     Tail   Hallux
## 1           1 400.2087 1234.1102 28.29390 230.7008 34.68051
## 2           2 189.3044  169.6962 12.04522 151.8225 15.87355
## 3           3 361.5988  942.8023 25.33677 215.5000 29.28605
```

```
aggregate(. ~ Cluster_B_2, data = Hawks[, c("Wing",
                                          "Weight",
                                          "Culmen",
                                          "Tail",
                                          "Hallux",
                                          "Cluster_B_2")], mean)
```

```
##   Cluster_B_2     Wing    Weight   Culmen     Tail   Hallux
## 1           1 195.2599  203.7531 12.71728 158.1265 16.68549
## 2           2 384.9118 1096.1076 27.00414 222.1887 31.97196
```

```
aggregate(. ~ Cluster_B_3, data = Hawks[, c("Wing", "Weight",
                                          "Culmen", "Tail",
                                          "Hallux", "Cluster_B_3")], mean)
```

```
##   Cluster_B_3     Wing    Weight   Culmen     Tail   Hallux
## 1           1 402.6651 1276.6459 28.31124 229.3014 35.33804
## 2           2 374.5475  990.7095 26.24106 218.0363 30.00684
## 3           3 195.2599  203.7531 12.71728 158.1265 16.68549
```

```
aggregate(. ~ Cluster_C_2, data = Hawks[, c("Wing", "Weight",
                                          "Culmen", "Tail",
                                          "Hallux", "Cluster_C_2")], mean)
```

```
##   Cluster_C_2     Wing    Weight   Culmen     Tail   Hallux
## 1           1 201.0722  227.6082 13.48814 160.8866 18.76031
## 2           2 384.9508 1158.7377 27.45246 214.4344 31.15697
```

```r
aggregate(. ~ Cluster_C_3, data = Hawks[, c("Wing", "Weight",
                                            "Culmen", "Tail", "Hallux",
                                            "Cluster_C_3")], mean)
```

```
##   Cluster_C_3     Wing    Weight   Culmen     Tail   Hallux
## 1           1 371.9481 1046.1169 26.65974 210.7922 30.37987
## 2           2 407.2000 1351.4444 28.80889 220.6667 32.48667
## 3           3 201.0722  227.6082 13.48814 160.8866 18.76031
```

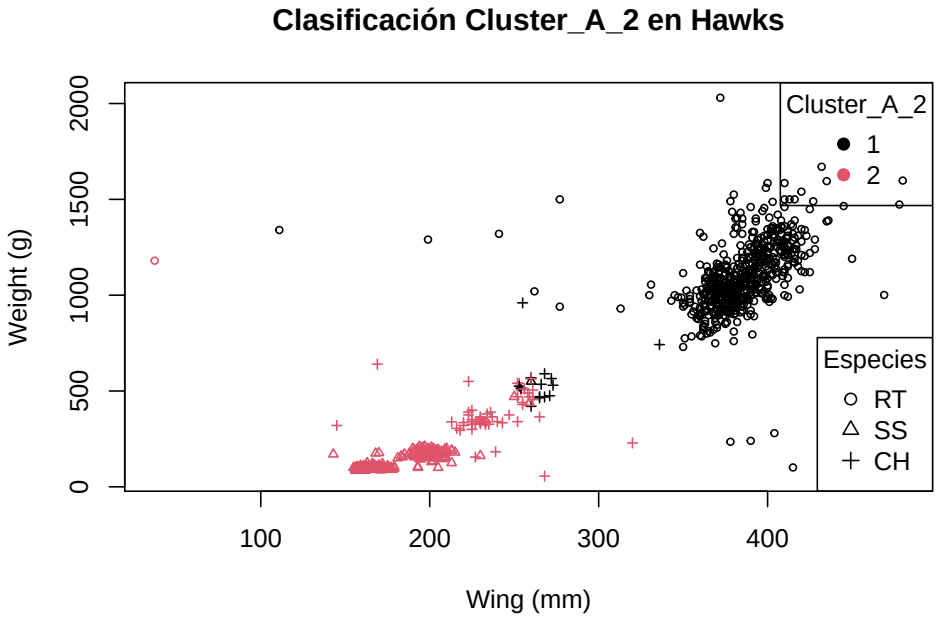
De los modelos probados, voy a comentar el A, he hecho el promedio, el cluster A para 2 clusters tiene claramente diferenciado las alas, el peso, etc, siendo unos halcones mucho mas grandes(1)que los otros(2). Si elijo el modelo con 3 clusters, se ve como tenemos uno con medidas muy grandes(1), otro con grandes(3), y otro con medidas pequeñas(2)

```r
# Conteo de registros por especie y cluster
conteo_A_3 <- as.data.frame(table(Hawks$Species, Hawks$Cluster_A_3))
colnames(conteo_A_3) <- c("Species", "Cluster_A_3", "Conteo_A_3")

conteo_A_2 <- as.data.frame(table(Hawks$Species, Hawks$Cluster_A_2))
colnames(conteo_A_2) <- c("Species", "Cluster_A_2", "Conteo_A_2")

conteo_combinado <- merge(conteo_A_2, conteo_A_3, by = "Species", all = TRUE)
conteo_combinado
```

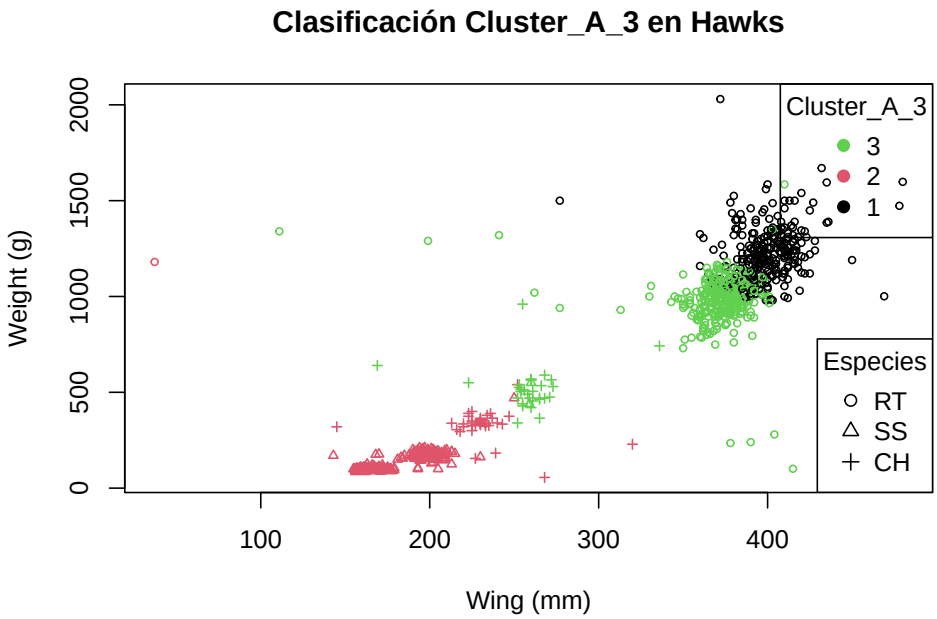
```
##    Species Cluster_A_2 Conteo_A_2 Cluster_A_3 Conteo_A_3
## 1       CH           1         16           1          0
## 2       CH           1         16           3         31
## 3       CH           1         16           2         38
## 4       CH           2         53           1          0
## 5       CH           2         53           3         31
## 6       CH           2         53           2         38
## 7       RT           1        565           1        254
## 8       RT           1        565           2          3
## 9       RT           1        565           3        311
## 10      RT           2          3           1        254
## 11      RT           2          3           2          3
## 12      RT           2          3           3        311
## 13      SS           1          2           2        252
## 14      SS           1          2           1          0
## 15      SS           1          2           3          3
## 16      SS           2        253           2        252
## 17      SS           2        253           1          0
## 18      SS           2        253           3          3
```

```r
plot_clusters <- function(data, cluster_column) {

  data_no_na <- subset(data, !is.na(data[[cluster_column]]))

  data_no_na$Species_numeric <- ifelse(data_no_na$Species == "RT", 1,
                                       ifelse(data_no_na$Species == "SS", 2,
                                              ifelse(data_no_na$Species == "CH", 3, NA)))
```

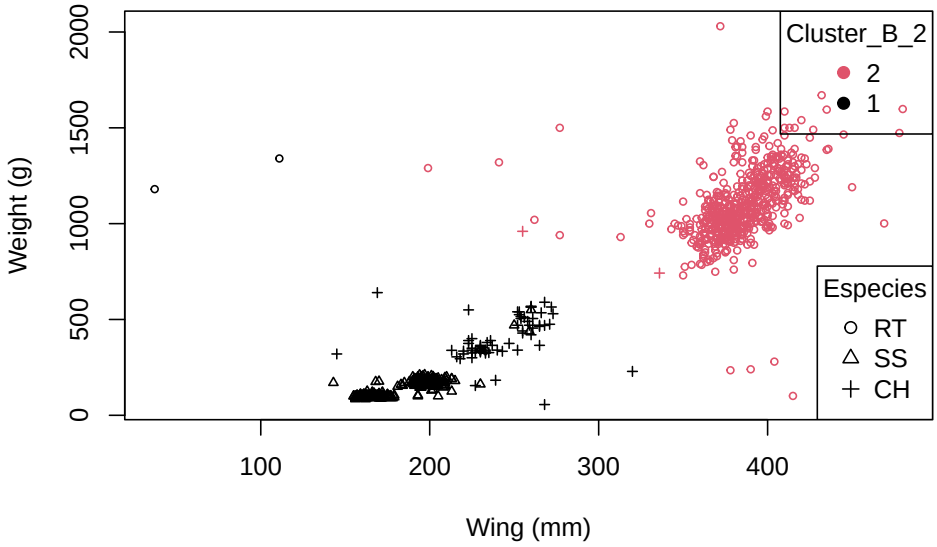
```
  plot(data_no_na$Wing, data_no_na$Weight,
       col = data_no_na[[cluster_column]],
       pch = data_no_na$Species_numeric,
       main = paste("Clasificación", cluster_column, "en Hawks"),
       xlab = "Wing (mm)",
       ylab = "Weight (g)",
       cex = 0.6)

  legend("topright",
         legend = unique(data_no_na[[cluster_column]]),
         col = unique(data_no_na[[cluster_column]]),
         pch = 19,
         title = cluster_column)

  legend("bottomright",
         legend = c("RT", "SS", "CH"),
         pch = c(1, 2, 3),
         title = "Especies")
}

plot_clusters(Hawks, "Cluster_A_2")
```



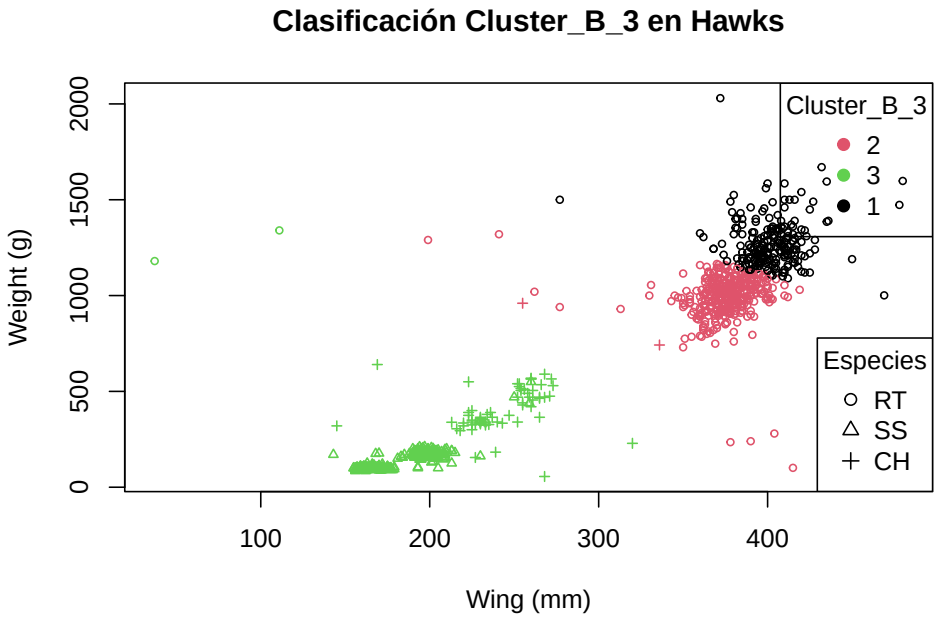

```
plot_clusters(Hawks, "Cluster_A_3")
```



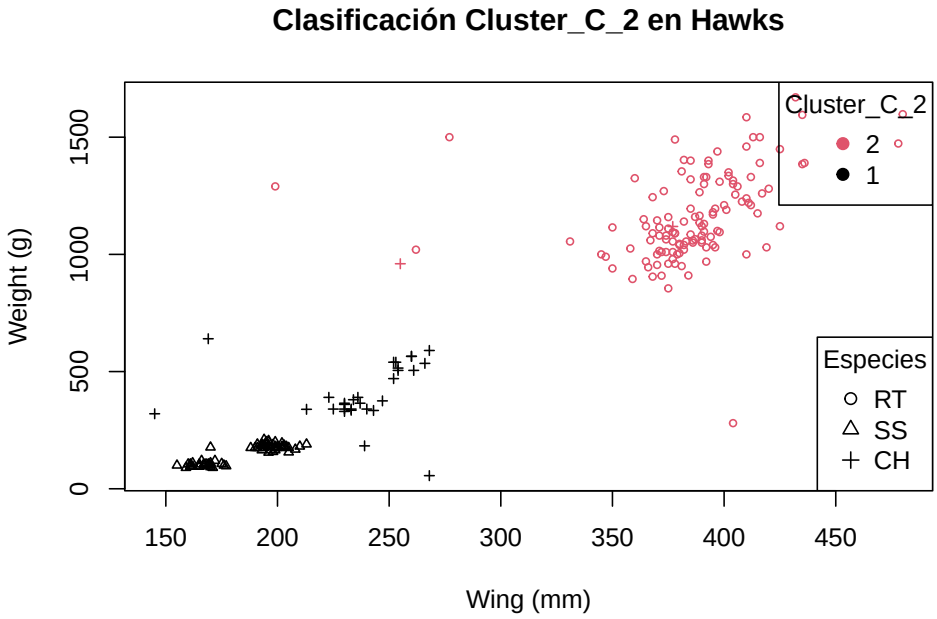

```
plot_clusters(Hawks, "Cluster_B_2")
```



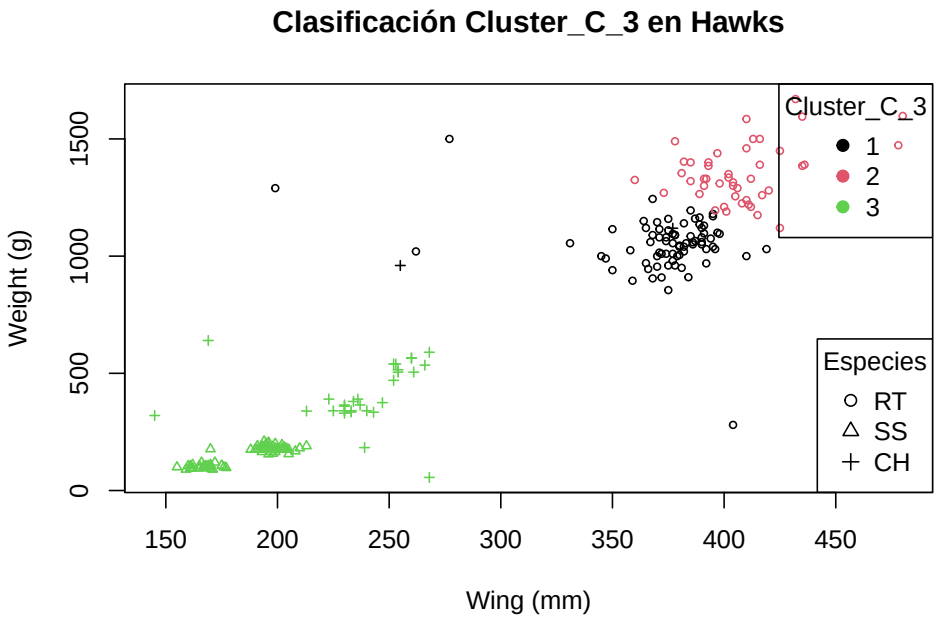
```
plot_clusters(Hawks, "Cluster_B_3")
```




```
plot_clusters(Hawks, "Cluster_C_2")
```




```
plot_clusters(Hawks, "Cluster_C_3")
```



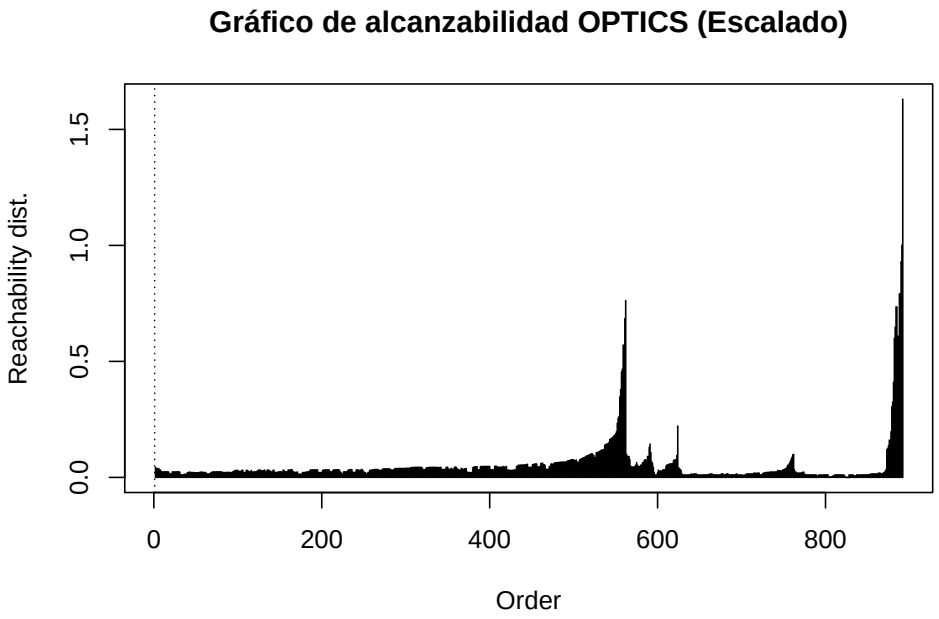
A excepcion de la pruebas de clusters A, B y C, con 3 clusters, el resto se asemeja bastante a las especies, aunque hay errores de asignación. Hay que tener en cuenta que en algunas ejecuciones me ha separado casi correctamente para algunos de los intentos de clusterización con 3 clusters, ya que la iniciacion de los centroides es aleatoria.

Ejercicio 2

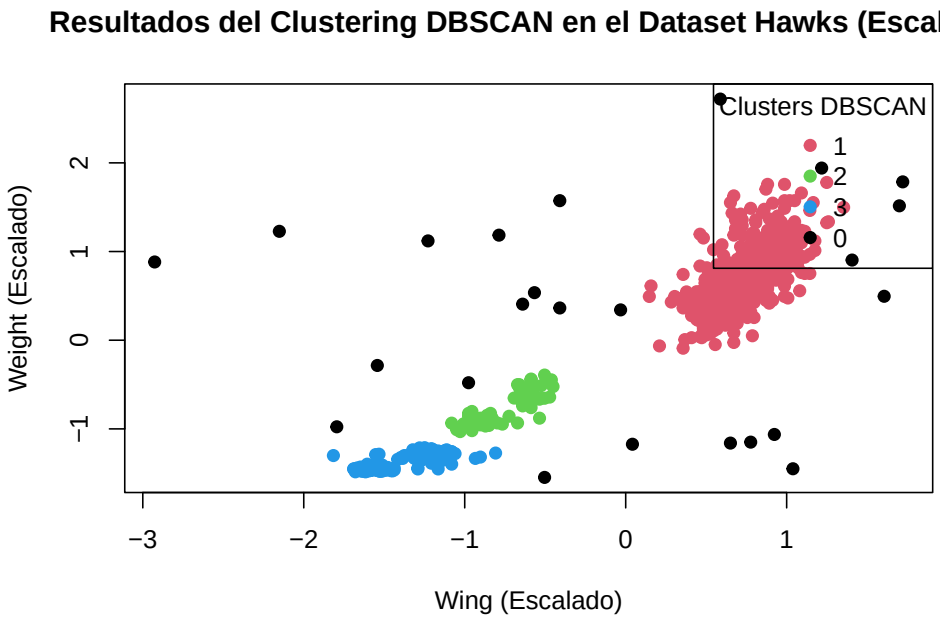
Con el juego de datos proporcionado realiza un estudio aplicando DBSCAN y OPTICS, similar al del ejemplo 2

Respuesta 2

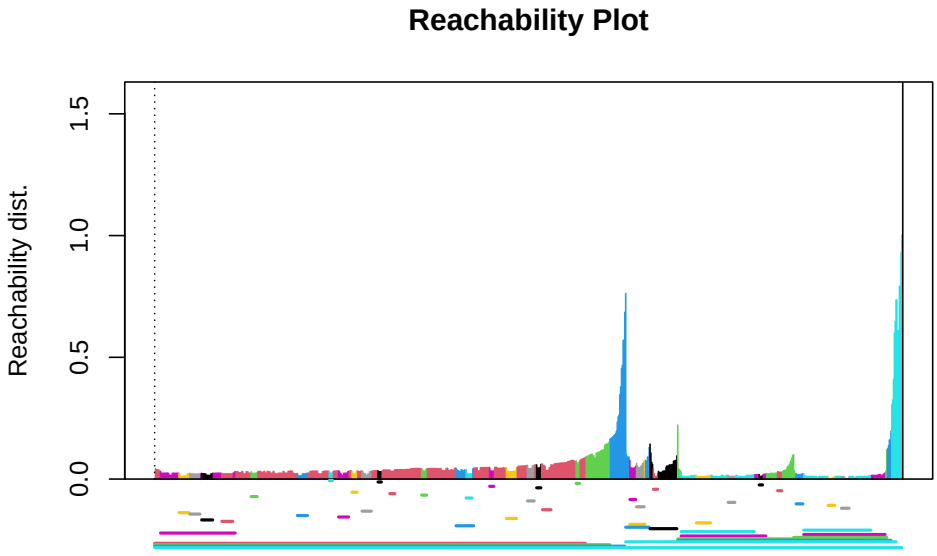
```
hawks_num_cluster <- na.omit(Hawks[, c("Wing", "Weight","Cluster_A_3")])
hawks_num <-hawks_num_cluster[, c("Wing", "Weight")]

hawks_num_scaled <- scale(hawks_num)
optics_result_scaled <- optics(hawks_num_scaled, minPts = 5)
plot(optics_result_scaled, main = "Gráfico de alcanzabilidad OPTICS (Escalado)")
```




```r
dbscan_extract_scaled <- extractDBSCAN(optics_result_scaled, eps_cl = 0.2)

plot(hawks_num_scaled[, 1], hawks_num_scaled[, 2],
     col = dbscan_extract_scaled$cluster + 1,
     main = "Resultados del Clustering DBSCAN en el Dataset Hawks (Escalado)",
     xlab = "Wing (Escalado)", ylab = "Weight (Escalado)", pch = 19)

legend("topright", legend = unique(dbscan_extract_scaled$cluster),
       col = unique(dbscan_extract_scaled$cluster + 1), pch = 19, title = "Clusters DBSCAN")
```




```
optics_result <- extractXi(optics_result_scaled, xi = 0.05)
plot(optics_result)
```



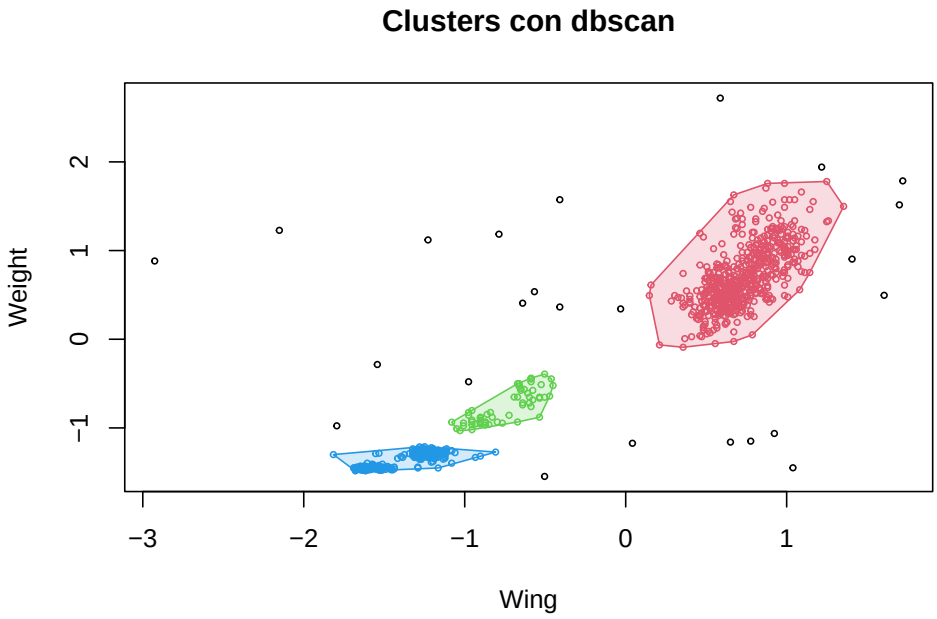
Si vemos el reachability plot o gráfico de alcanzabilidad en español, hay unos picos muy altos, que representan la densidad, puesto que DBSCAN se basa en la densidad, y si hay clusters con diferentes densidades, quizás no lo detecte bien. Como vemos aquí, en general hay una densidad alta, con picos, que son outliers en muchos casos, estos picos representan zonas de baja densidad.

DBSCAN crea clusters en base a la densidad con dos variables fundamentales, Eps y MinPts. Eps representa la distancia maxima entre puntos vecinos y MinPts el numero minimo de puntos para formar un cluster.. dicho esto, cuando se utiliza DBSCAN hay que hacer atención a configurar la densidad adecuadamente, igual que en Kmeans se ce crean los centroides y luego se asignan los valores a su centroide mas cercano, generando esferas circulares.

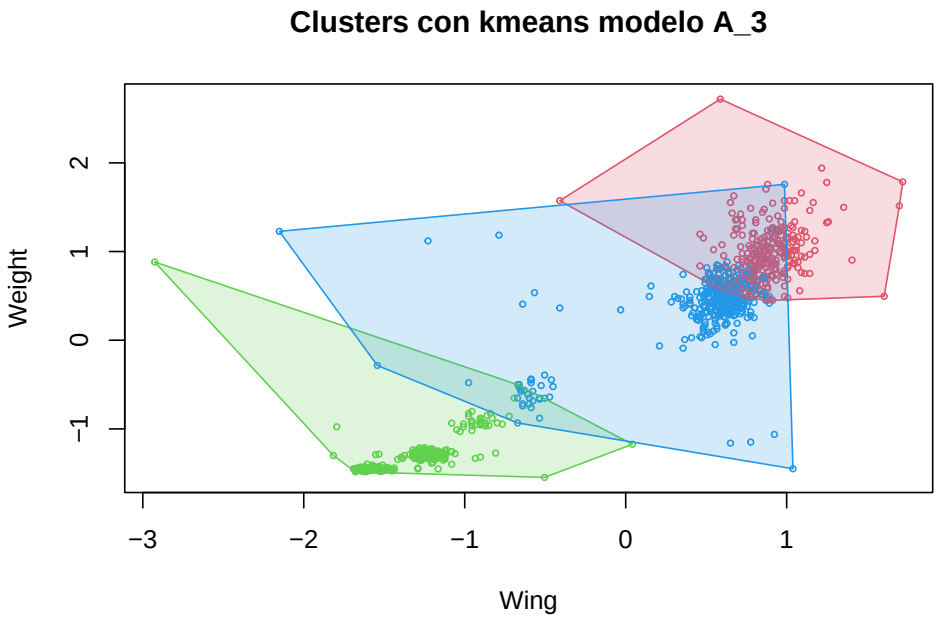
Ejercicio 3

Realiza una comparativa de los métodos *k-means* y *DBSCAN*

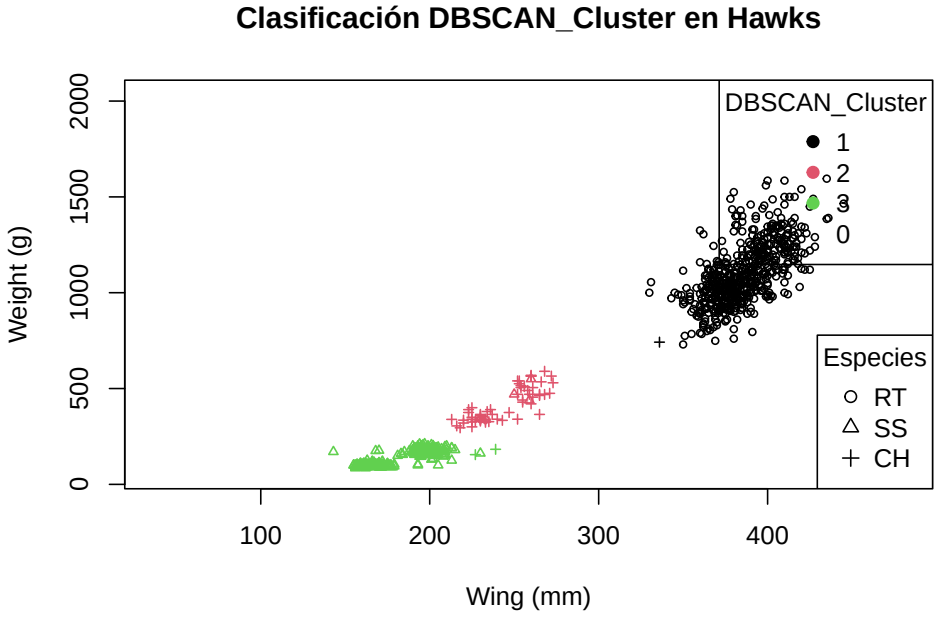
```
hullplot(hawks_num_scaled, dbscan_extract_scaled$cluster, main = "Clusters con dbscan")
```




```
hullplot(hawks_num_scaled,hawks_num_cluster$Cluster_A_3 , main = "Clusters con kmeans modelo A_3")
```




```r
hawks_num_cluster$DBSCAN_Cluster <- dbscan_extract_scaled$cluster
Hawks$DBSCAN_Cluster <- NA  # Crear una columna en Hawks para los clusters
Hawks[rownames(hawks_num_cluster), "DBSCAN_Cluster"] <- hawks_num_cluster$DBSCAN_Cluster

plot_clusters(Hawks, "DBSCAN_Cluster")
```



Como conclusion, DBSCAN es la eleccion adecuada ya que tiene menos ruido, y los valores outliers o atipicos no generan un problema por su baja densidad y ayuda a detectar mejor los clusters. kmeans es interesante

siempre (ya que es un modelo muy utilizado y útil) y es mas interesante en datos que tienen una estructura mas simple, y si son datos circulares su distribución al centroide, mejor aún. Por tanto, yo creo que DBSCAN captura mejor la estructura de datos, viendo el ejercicio se ve como he probado muchas combinaciones para kmeans, mas las que borré y reescribí, sin embargo con DBSCAN ha sido mucho más rápido encontrar una distribución de clusters adecuada y funciona mucho mejor. Además no funciona de forma aleatoria como en kmeans la selección de los centroides.Por tanto DBSCAN es claramente mucho mas robusto en este dataset.

Referencias: Laboratorio en Python y R. Mineria de datos. datacamp.com openclassrooms.com