Universitat Oberta de Catalunya

Estudios de Informática, Multimedia y Telecomunicaciones

Minería de datos: PEC2 - Métodos no supervisados

Gloria Manresa

Mayo 2023

0.1 Ejercicio 1

0.1.1 Enunciado

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

Adicionalmente realiza un estudio similar al de los ejemplos 1.1 y 1.2.

- 10%. Se explican los campos de la base de datos
- 25%. Se aplica el algoritmo de k-means de forma correcta.
- 25%. Se prueban con diferentes valores de k.
- 10%. Se obtiene una medida de lo bueno que es el agrupamiento.
- 20%. Se describen e interpretan los diferentes clusters obtenidos.
- 10%. Se presenta el código y es fácilmente reproducible.

0.1.2 Descripción del origen del conjunto de los datos.

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.

0.1.3 Carga de los datos

En un primer lugar cargamos los datos.

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

A continuación presentamos los datos con el siguiente resumen informativo, dónde podemos ver los valores que toman cada una de las variables.

```
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
```

Observamos que el juego de datos presenta 19 columnas y 908 observaciones. Los nombres y significados de las columnas son los siguientes:

- Month: Mes, toma valores desde 8 a 11.
- Day: Día del mes.
- Year: Año, toma valores desde 1992 hasta 2003.
- CaptureTime: Hora de la captura.
- ReleaseTime: Hora de la liberación.
- BandNumber: Identificación.
- Species: Código de una de las tres especies (CH=Cooper's, RT=Red-tailed, SS=Sharp-Shinned).
- Age: Toma los valores A = Adult o I = Imature.
- Sex: Toma los valores F = Female o M = Male.
- Wing: Longitud (en mm) de la pluma principal del ala desde la punta hasta la muñeca a la que se une.
- Weight: Peso corporal (en gramos).
- Culmen: Longitud (en mm) del pico superior desde la punta hasta donde choca con la parte carnosa del ave.
- Hallux: Longitud (en mm) de la garra.
- Tail: Medida (en mm) relacionada con la longitud de la cola (inventada en el MacBride Raptor Center)
- StandardTail: Medida estándar de la longitud de la cola (en mm)

- Tarsus: Longitud del hueso básico de la pata (en mm).
- WingPitFat: Cantidad de grasa en la ala.
- KeelFat: Cantidad de grasa en el esternón (medida al tacto).
- Crop: Cantidad de material en el cultivo, codificado de 1=lleno a 0=vacío.

0.1.4 Limpieza de los datos

Las variables numéricas en las que se basará el siguiente estudio son: Wing, Weight, Culmen, Hallux. A continuación mostramos el porcentaje de valores NA que tienen dichas variables:

```
# Porcentaje valores NA
sort(colMeans(is.na(Hawks[,10:13])), decreasing = TRUE)*100
```

```
##    Weight    Culmen    Hallux      Wing
## 1.1013216 0.7709251 0.6607930 0.1101322
```

```
# Registros valores NA
colSums(is.na(Hawks[,10:13]))
```

```
##   Wing Weight Culmen Hallux
##      1     10      7      6
```

Dado que los porcentajes son muy bajos, menos del 1'2% se decide borrarlos.

Guardamos en una nueva variable el subconjunto de los datos que incluya únicamente las cuatro variables numéricas con las que vamos a trabajar y elimina los registros que contienen Na's.

```
x <- na.omit(Hawks[,10:13])
```

Antes de proceder con el estudio escalamos los datos de las variables a utilizar para que todas tengan el mismo peso en el algoritmo.

```
x_scale <- scale(x)
```

0.1.5 Método de agregación k-means

A continuación vamos a utilizar el método de agregación k-means sobre el data set "Hawks" presentado anteriormente.

El objetivo del ejercicio es encontrar agrupaciones para predecir la columna "Species". Se realizará con el método no supervisado de agregación k-means por lo que no se utilizará dicha columna en el estudio.

Trabajaremos únicamente con las variables numéricas: Wing, Weight, Culmen, Hallux.

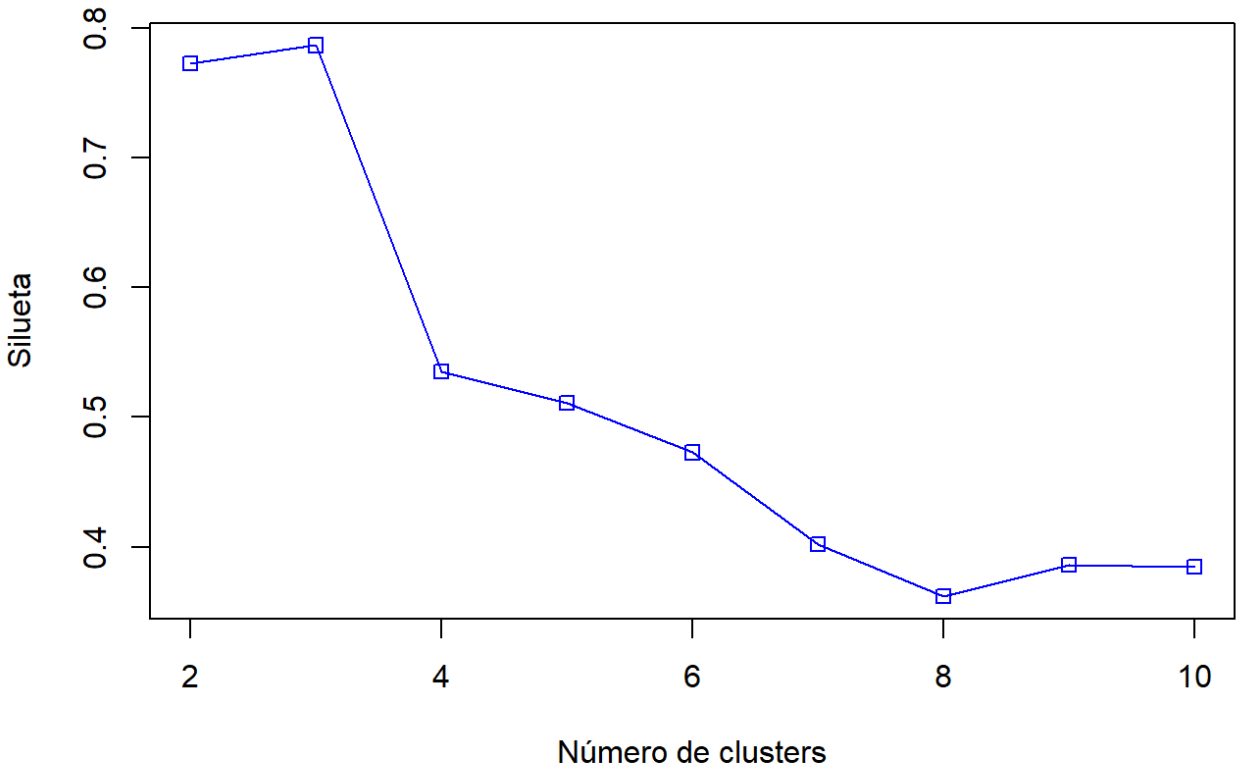
Hay que tener en cuenta que teóricamente no conocemos el número óptimo de clústers por lo que probaremos con varios valores.

```
library(cluster)
set.seed(3)

d <- daisy(x_scale)
resultados <- rep(0, 10)
for (i in c(2,3,4,5,6,7,8,9,10))
{
  fit           <- kmeans(x_scale, i)
  y_cluster     <- fit$cluster
  sk            <- silhouette(y_cluster, d)
  resultados[i] <- mean(sk[,3])
}

resultados[2:10]
```

```
## [1] 0.7728399 0.7870581 0.5353183 0.5110736 0.4727849 0.4018636 0.3616606
## [8] 0.3861249 0.3843435
```

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

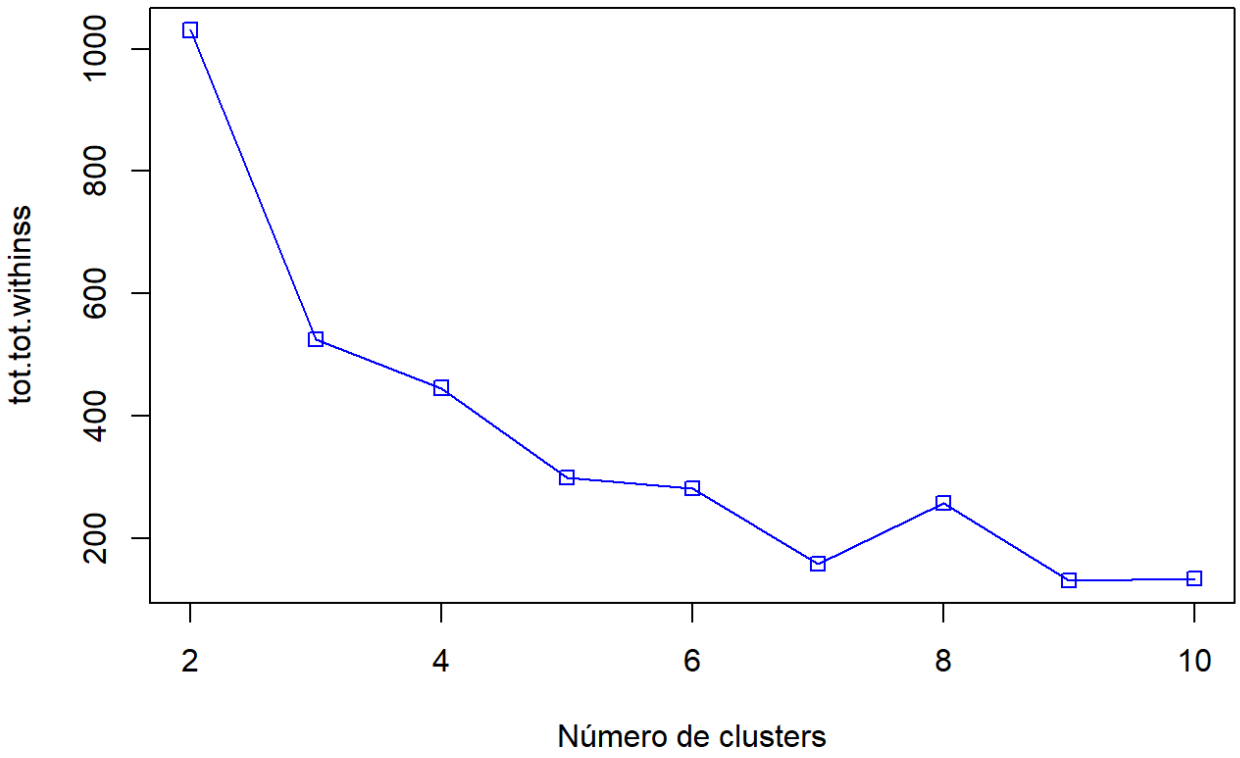
```
plot(2:10,resultados[2:10],type="o",col="blue",pch=0,xlab="Número de clusters",ylab="Sil
ueta")
```



Según los resultados obtenidos a partir de este método observamos que el número de clusteres óptimo parece ser 3 y muy de cerca le siguen 2 clusteres. Pasamos a realizar otra prueba para comparar los resultados.

Aplicamos a continuación el método del codo.

```
set.seed(3)
resultados <- rep(0, 10)
for (i in c(2,3,4,5,6,7,8,9,10))
{
  fit           <- kmeans(x_scale, i)
  resultados[i] <- fit$tot.withinss
}
plot(2:10,resultados[2:10],type="o",col="blue",pch=0,xlab="Número de clusters",ylab="to
t.tot.withinss")
```



En este caso, coincidiendo con el método anterior, el número óptimo de clústers es 3 que es cuando la curva comienza a estabilizarse.

Continuamos con el estudio del valor óptimo de clusters. En este caso sabemos que se trata de 3 clusters pero lo habitual es no conocer este número.

Utilizaremos a continuación la función kmeansruns 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)

set.seed(3)
fit_ch  <- kmeansruns(x_scale, krange = 1:10, criterion = "ch")
fit_asw <- kmeansruns(x_scale, krange = 1:10, criterion = "asw")
```

Mostramos el resultado obtenido para todos los valores de k usando ambos criterios:

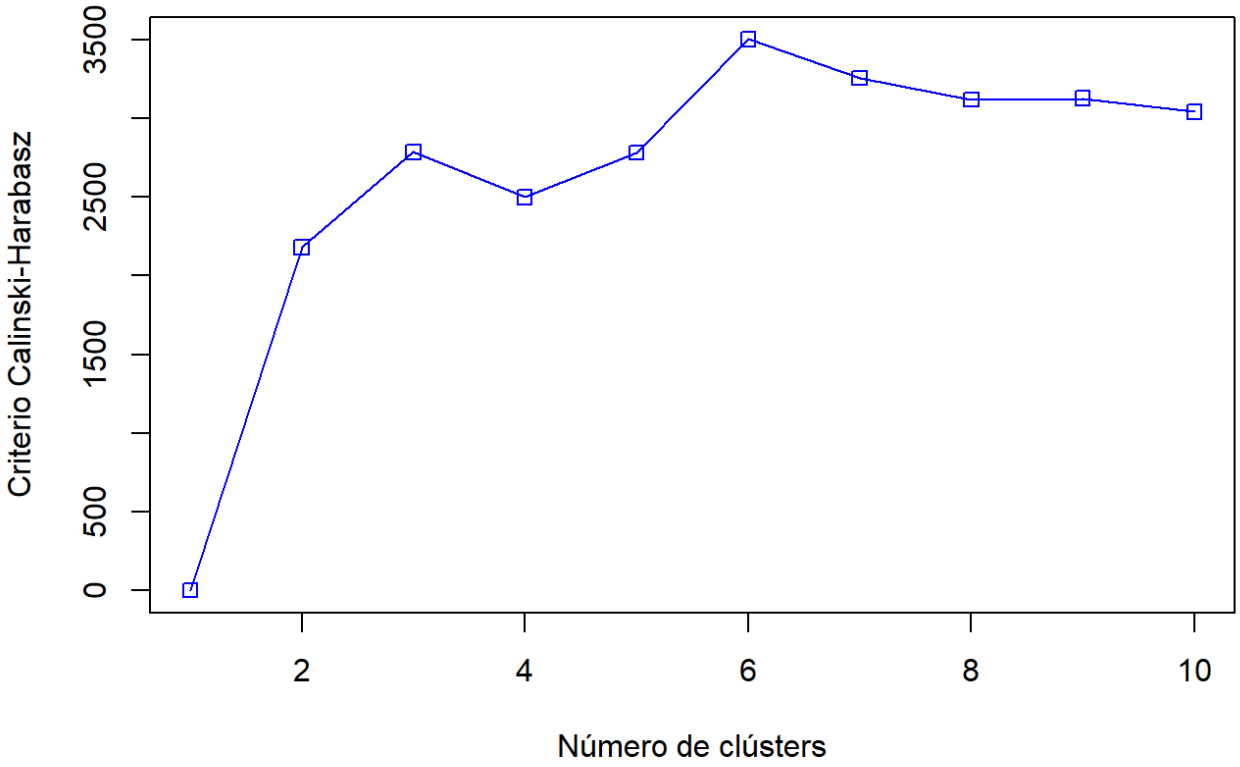
```
fit_ch$bestk
```

```
## [1] 6
```

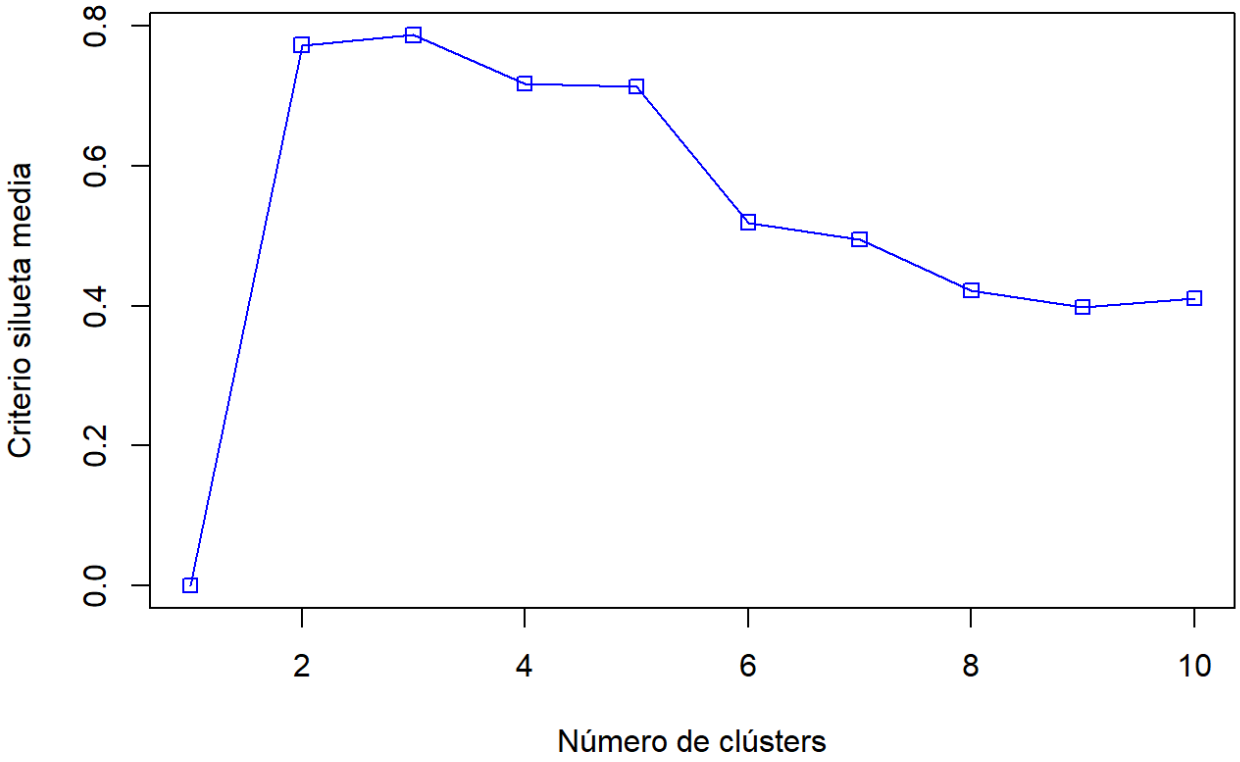
```
fit_asw$bestk
```

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

```
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")
```

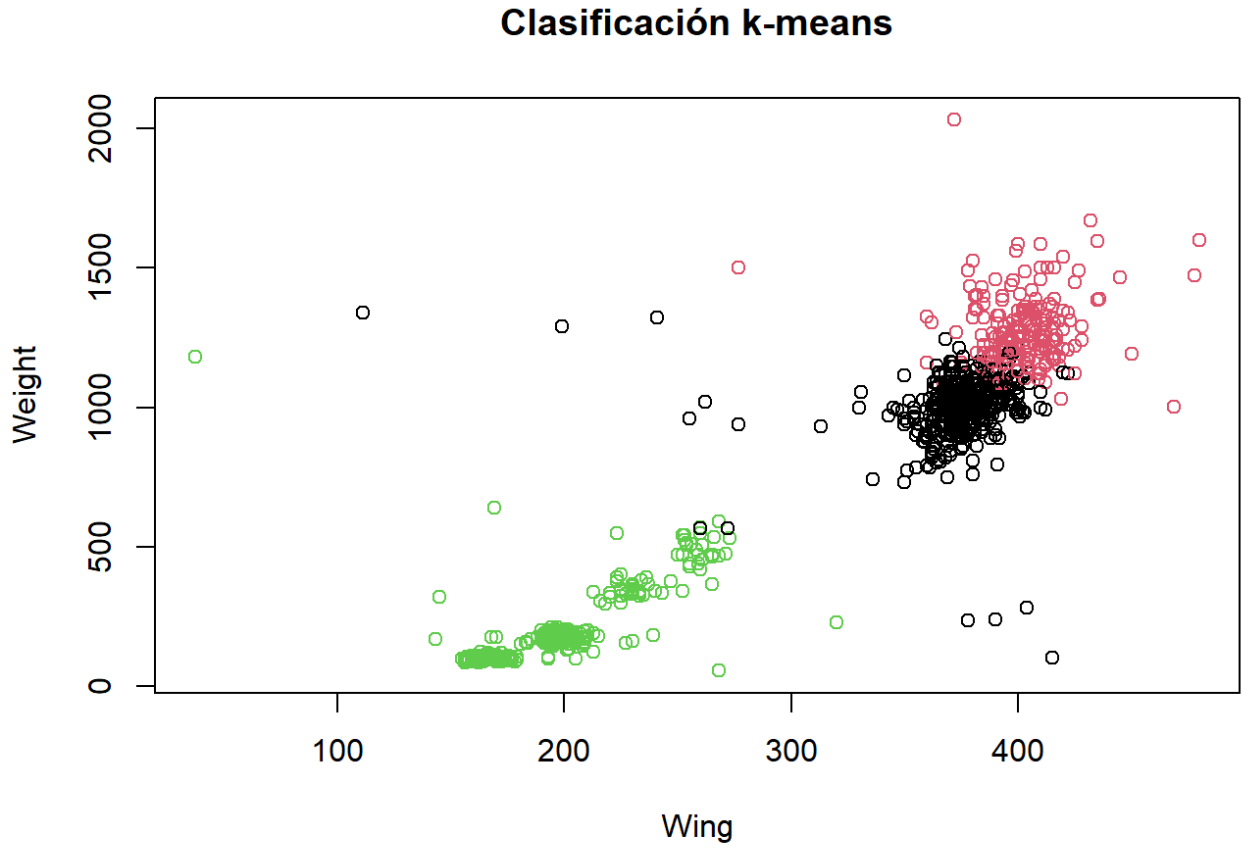


Con el criterio de la silueta media se obtienen tres clústers y con el criterio de Calinski-Harabasz se obtienen 6.

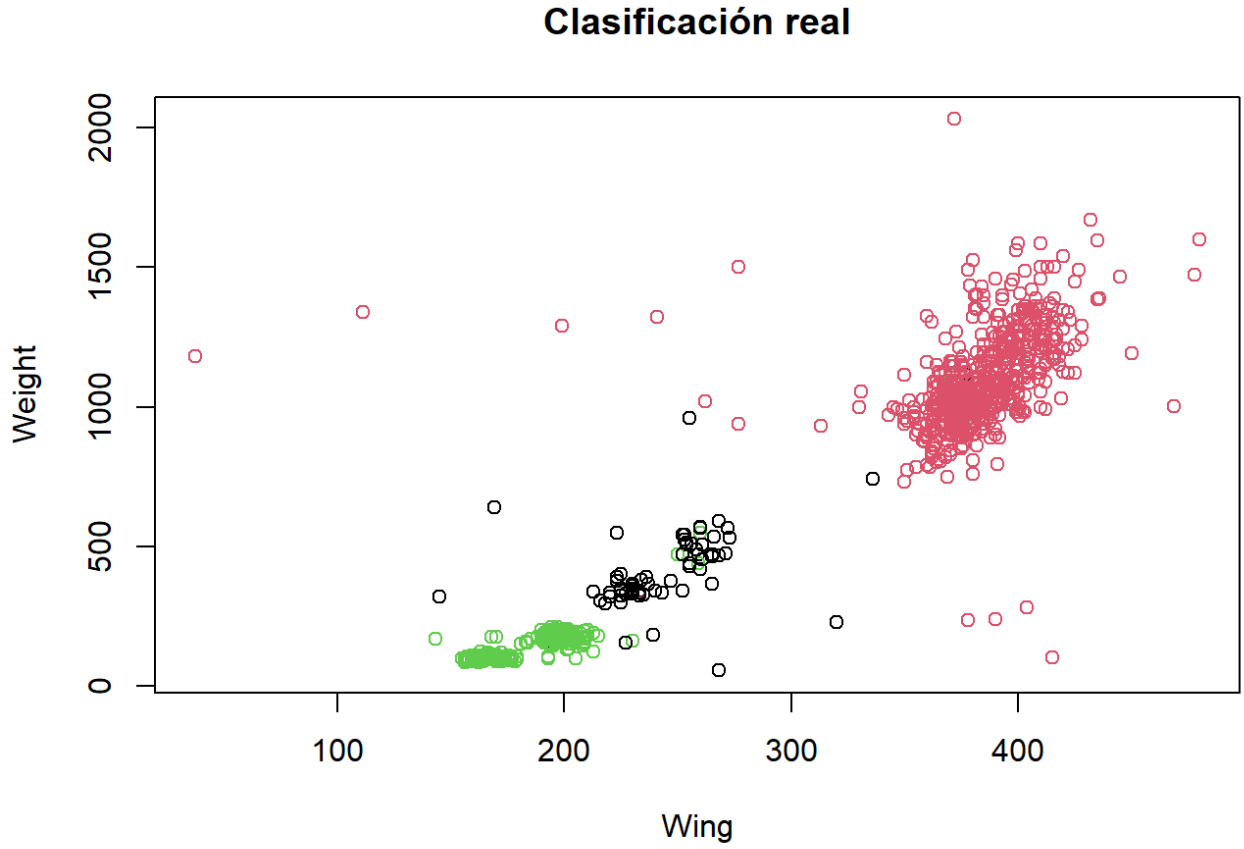
En este caso y puesto que de los métodos utilizados tres clústers ha sido el valor óptimo en la mayoría de ellos optaremos por tomar este valor para realizar nuestro algoritmo de kmeans.

A continuación estudiamos 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.

```
set.seed(3)
Hawks3clusters <- kmeans(x_scale, 3)

# Weight y Wing
plot(x[c(1,2)], col=Hawks3clusters$cluster, main="Clasificación k-means")
```

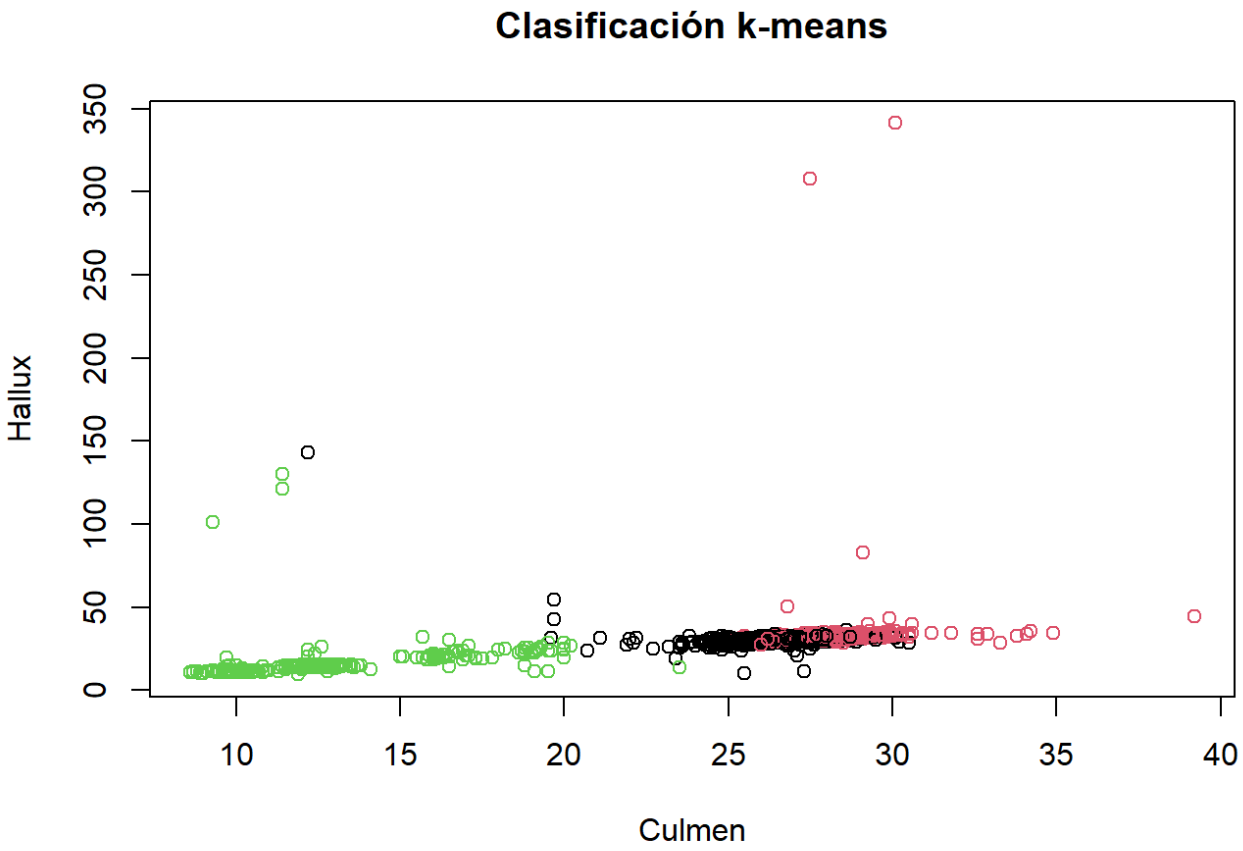



```
plot(Hawks[c(10,11)], col=as.factor(Hawks$Species), main="Clasificación real")
```

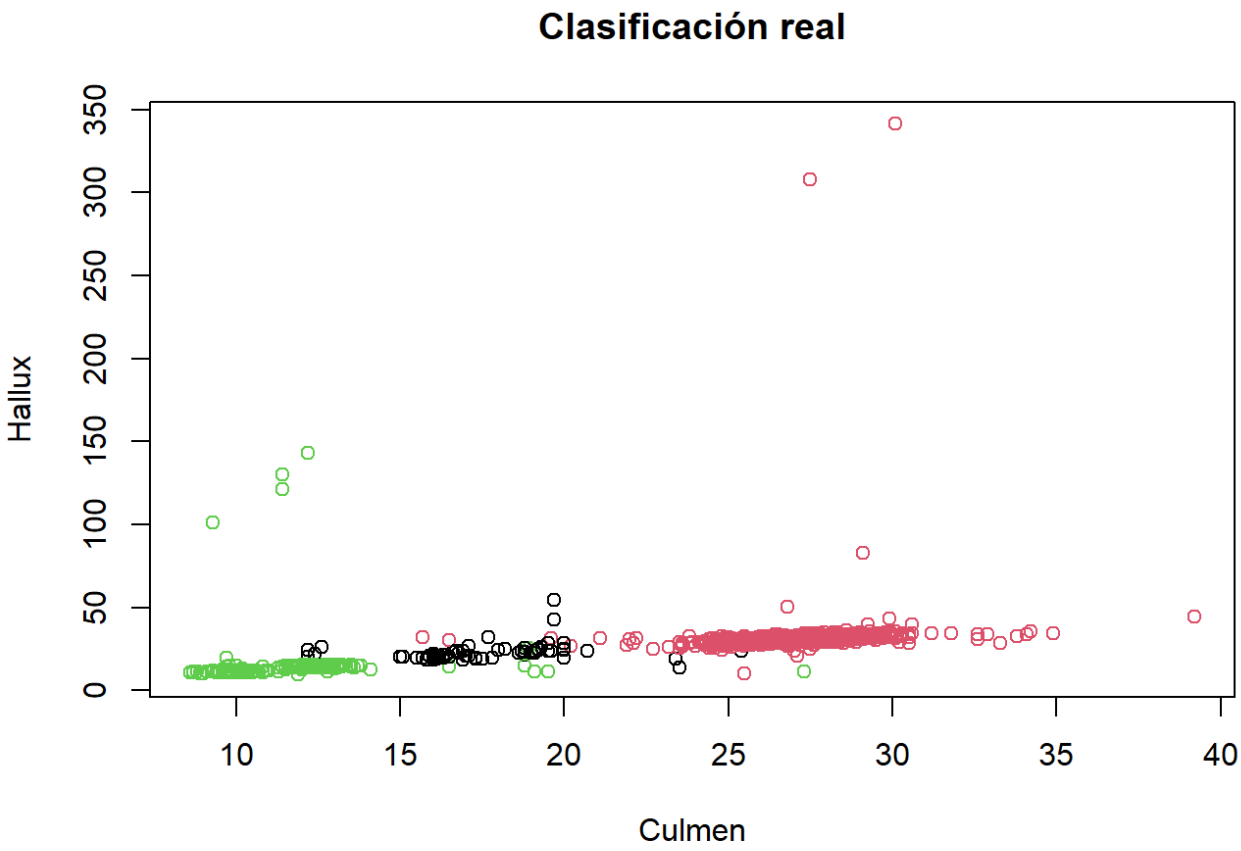


Observamos que a pesar que “Weight” y “Wing” podrían ser buenos indicadores para diferenciar a las tres especies, el algoritmo de kmeans confunde una de las especies asignándola incorrectamente.

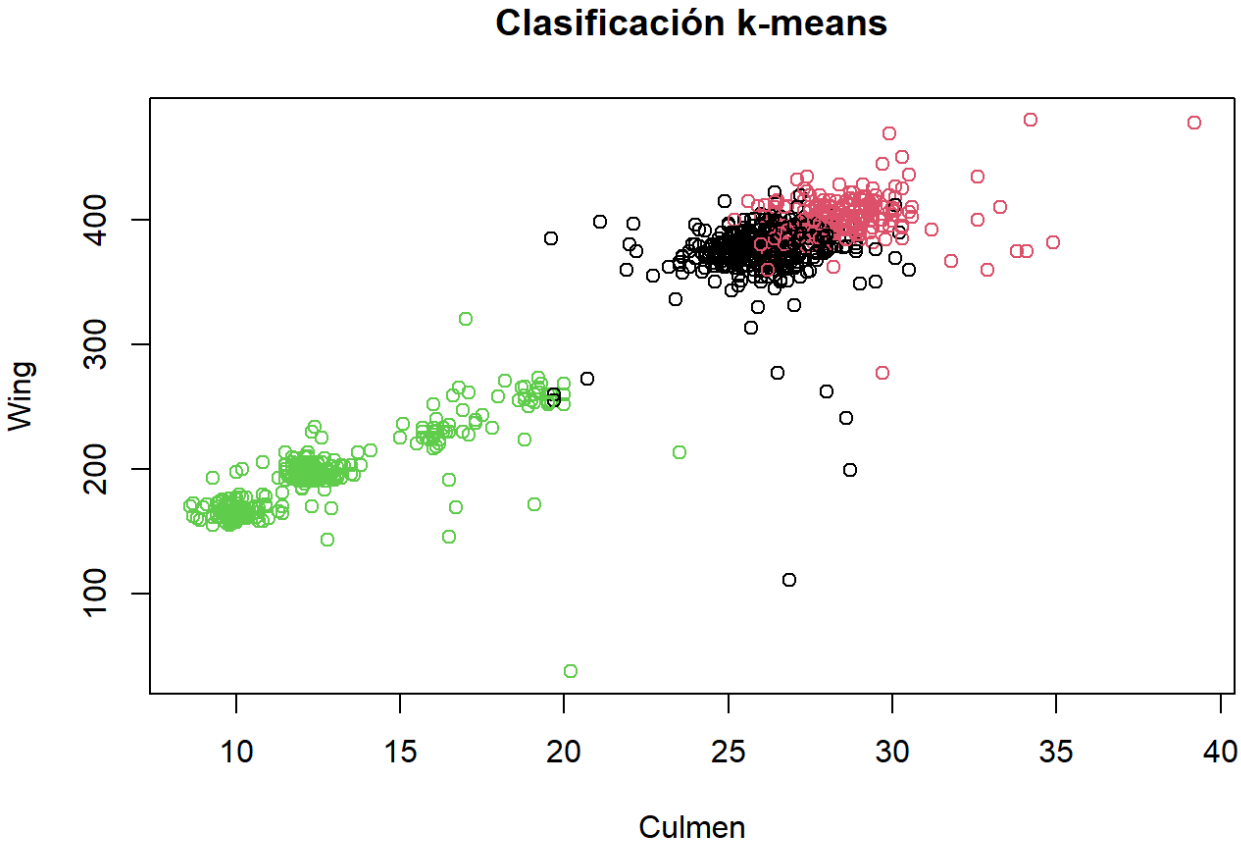
```
# Hallux y Culmen
plot(x[c(3,4)], col=Hawks3clusters$cluster, main="Clasificación k-means")
```



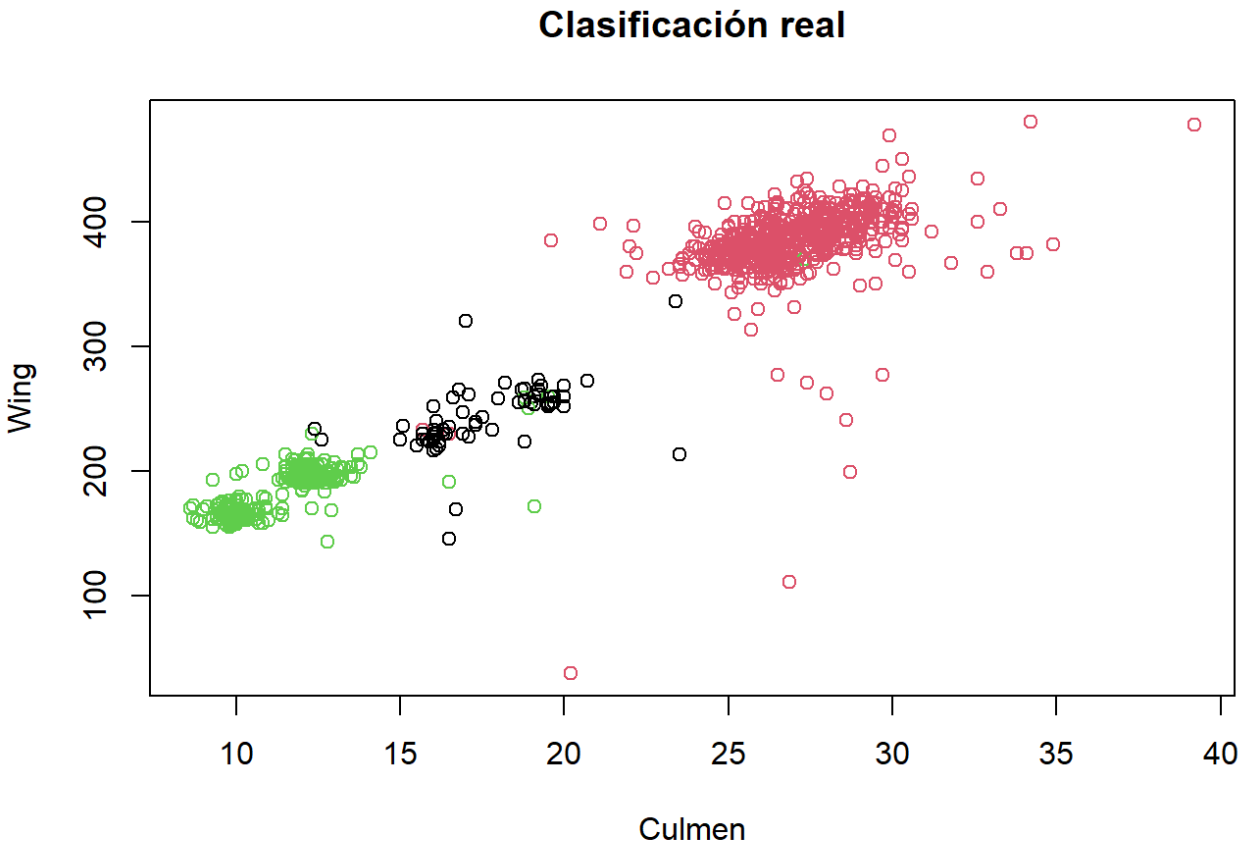

```
plot(Hawks[c(12,13)], col=as.factor(Hawks$Species), main="Clasificación real")
```



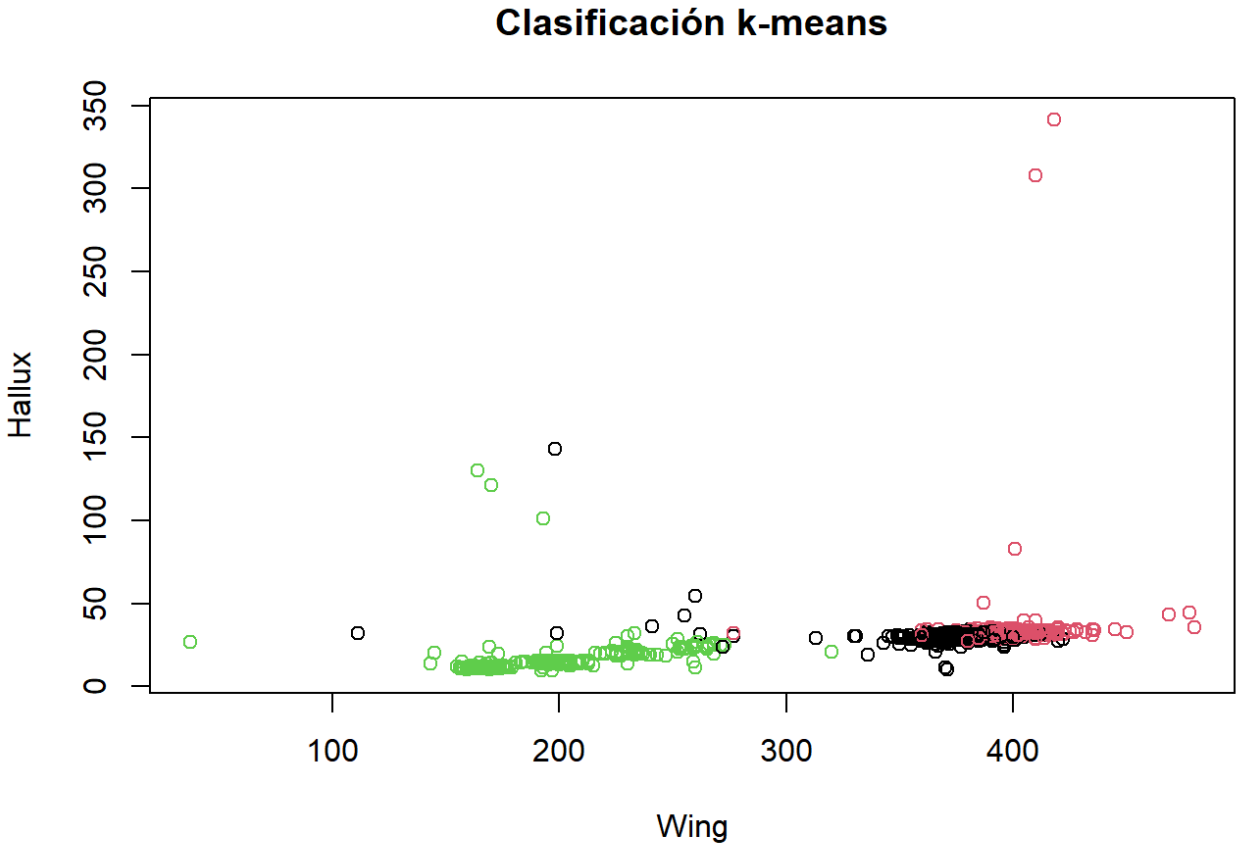

```
# Culmen y Wing
plot(x[c(3,1)], col=Hawks3clusters$cluster, main="Clasificación k-means")
```



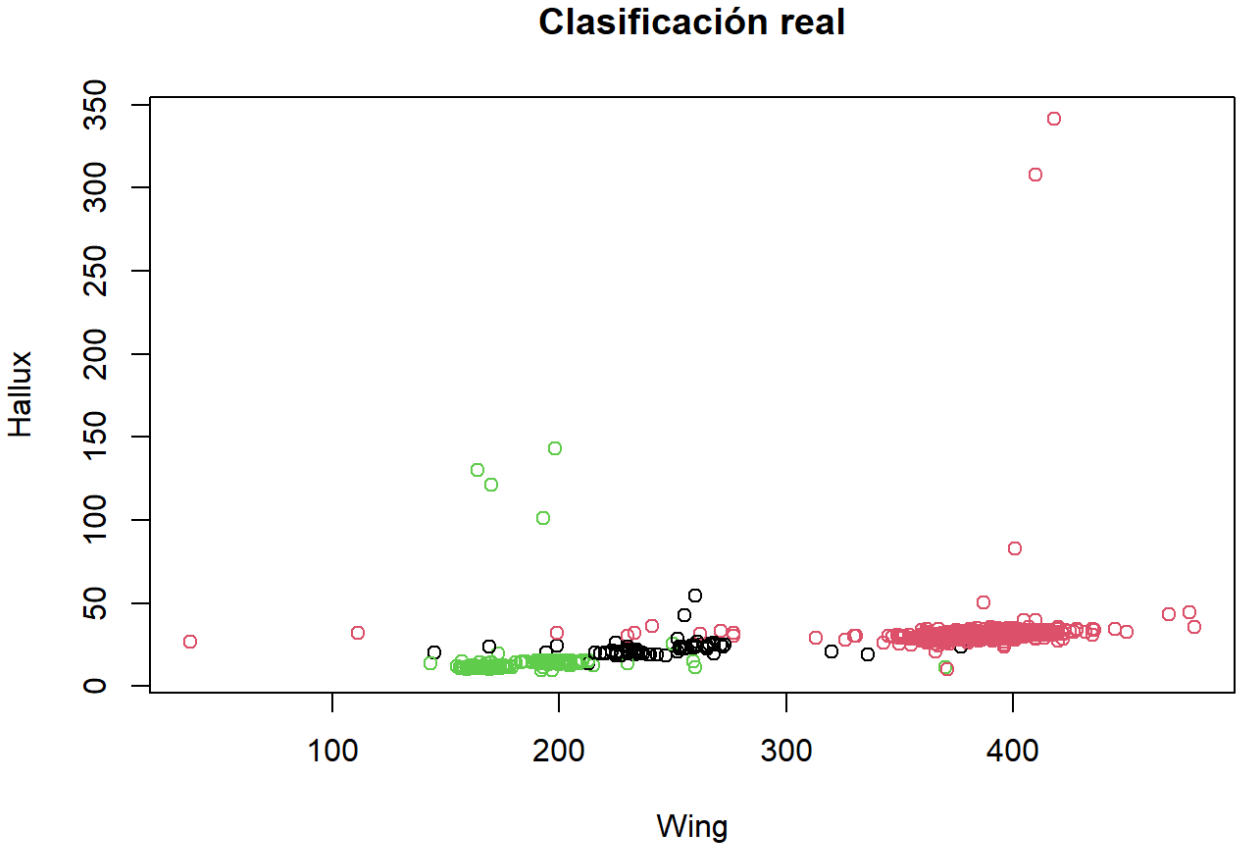
```
plot(Hawks[c(12,10)], col=as.factor(Hawks$Species), main="Clasificación real")
```



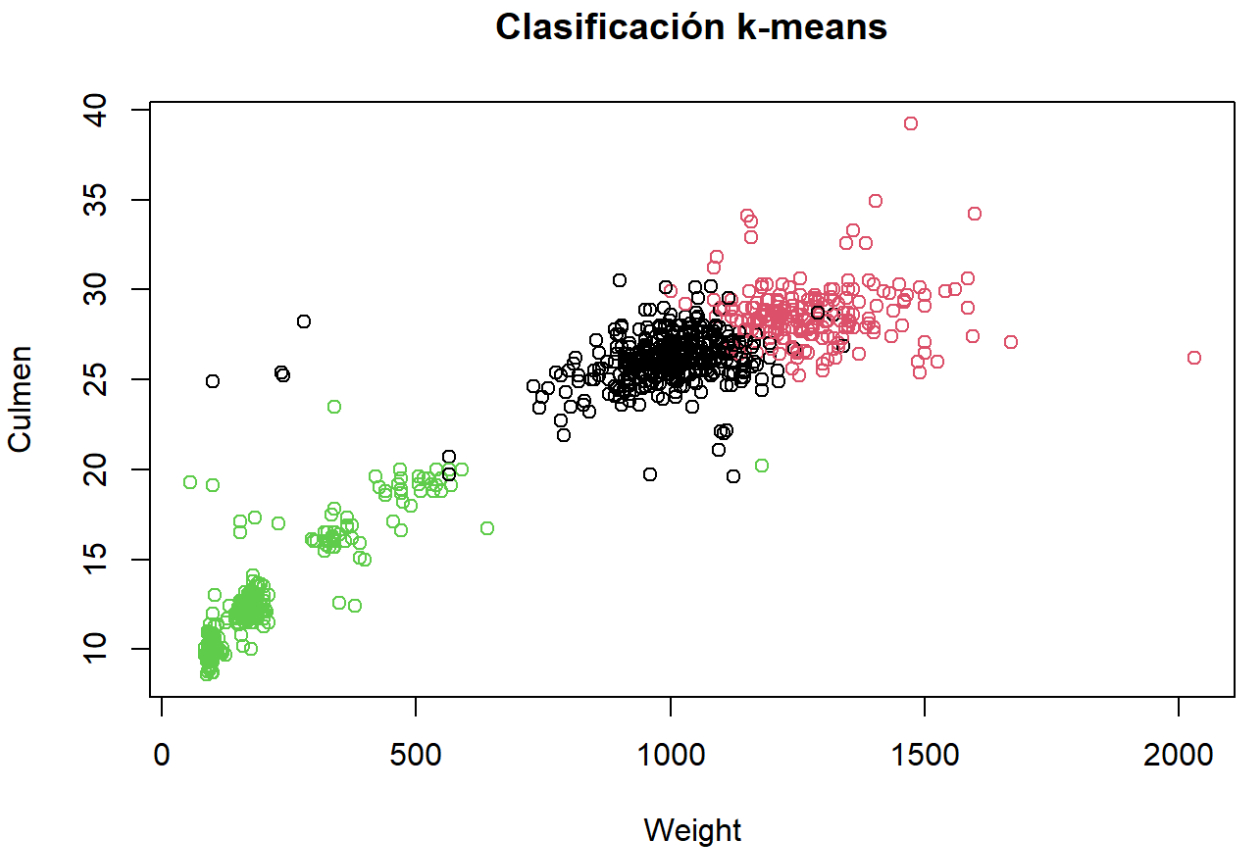

```
# Hallux y Wing
plot(x[c(1,4)], col=Hawks3clusters$cluster, main="Clasificación k-means")
```



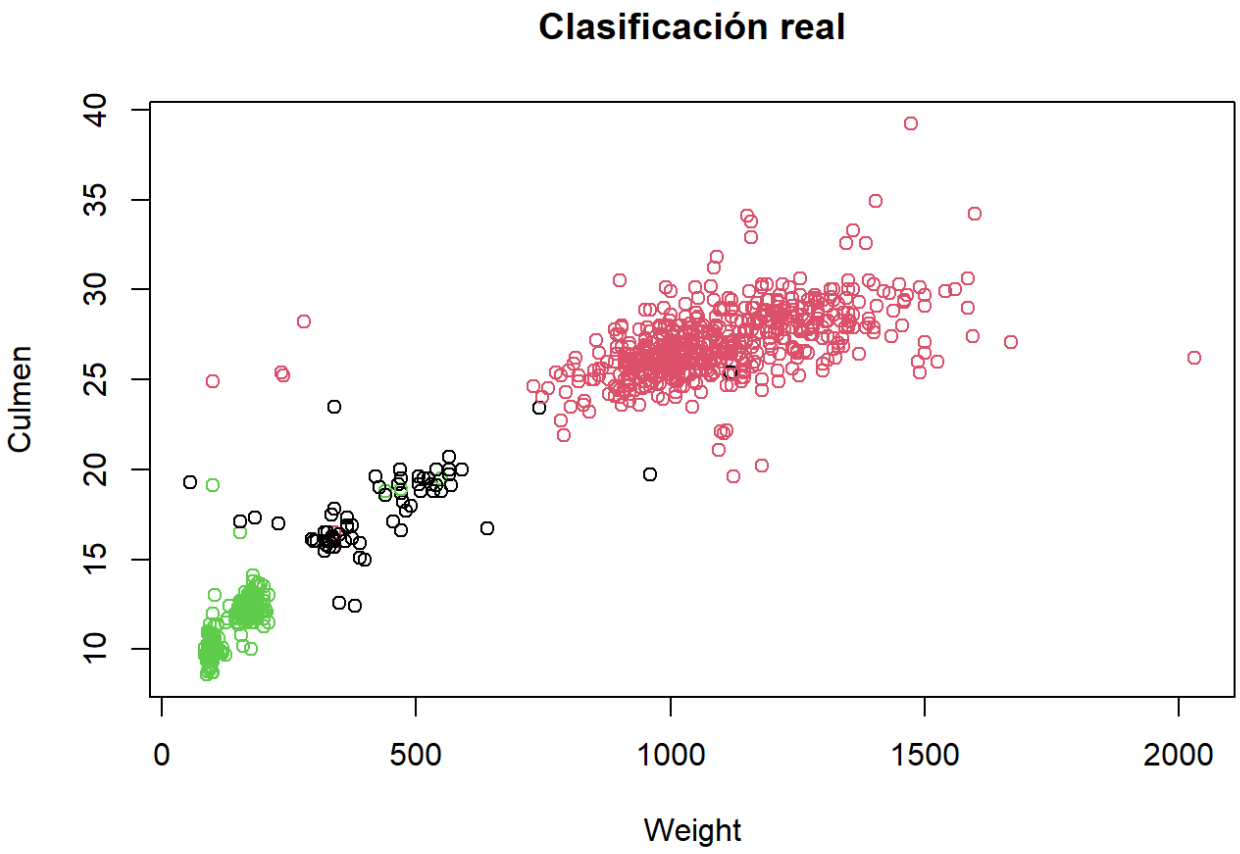

```
plot(Hawks[c(10,13)], col=as.factor(Hawks$Species), main="Clasificación real")
```



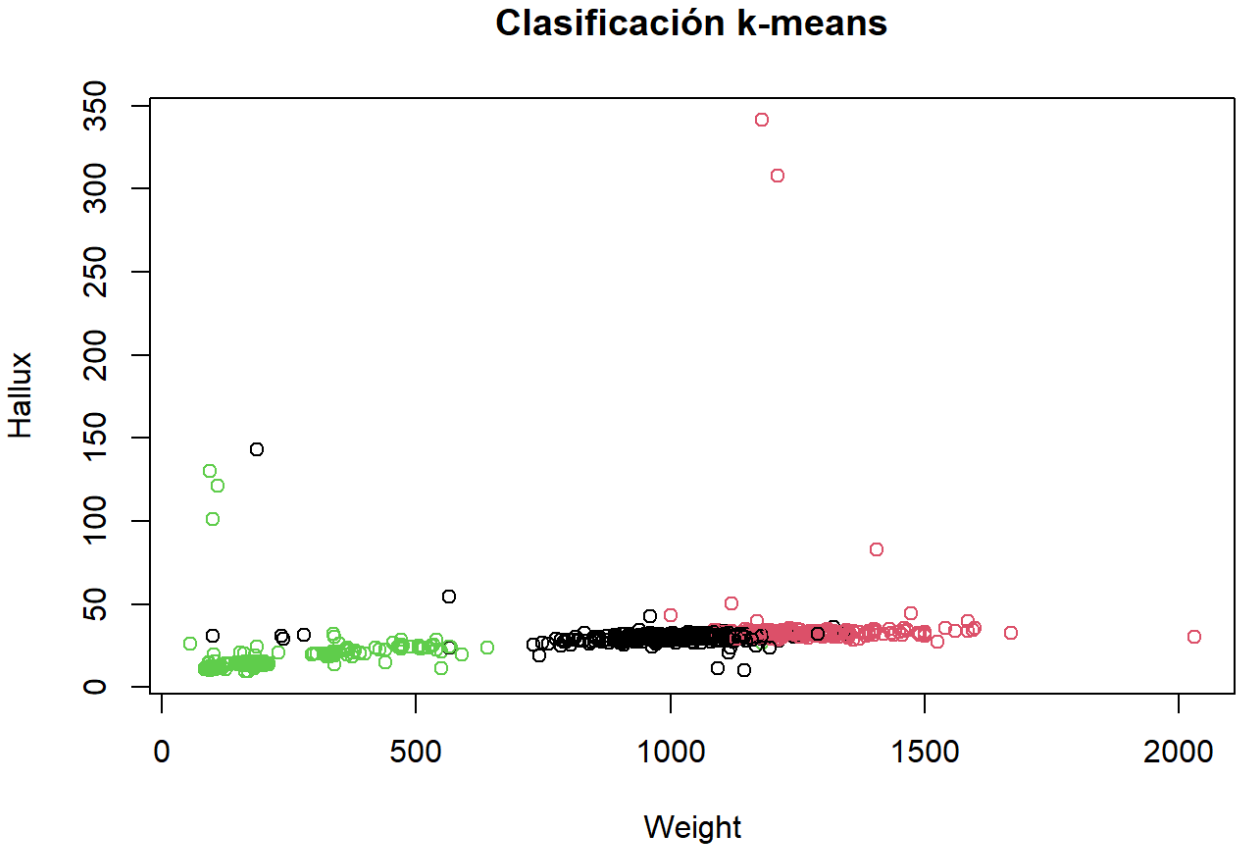
```
# Culmen y Weight
plot(x[c(2,3)], col=Hawks3clusters$cluster, main="Clasificación k-means")
```



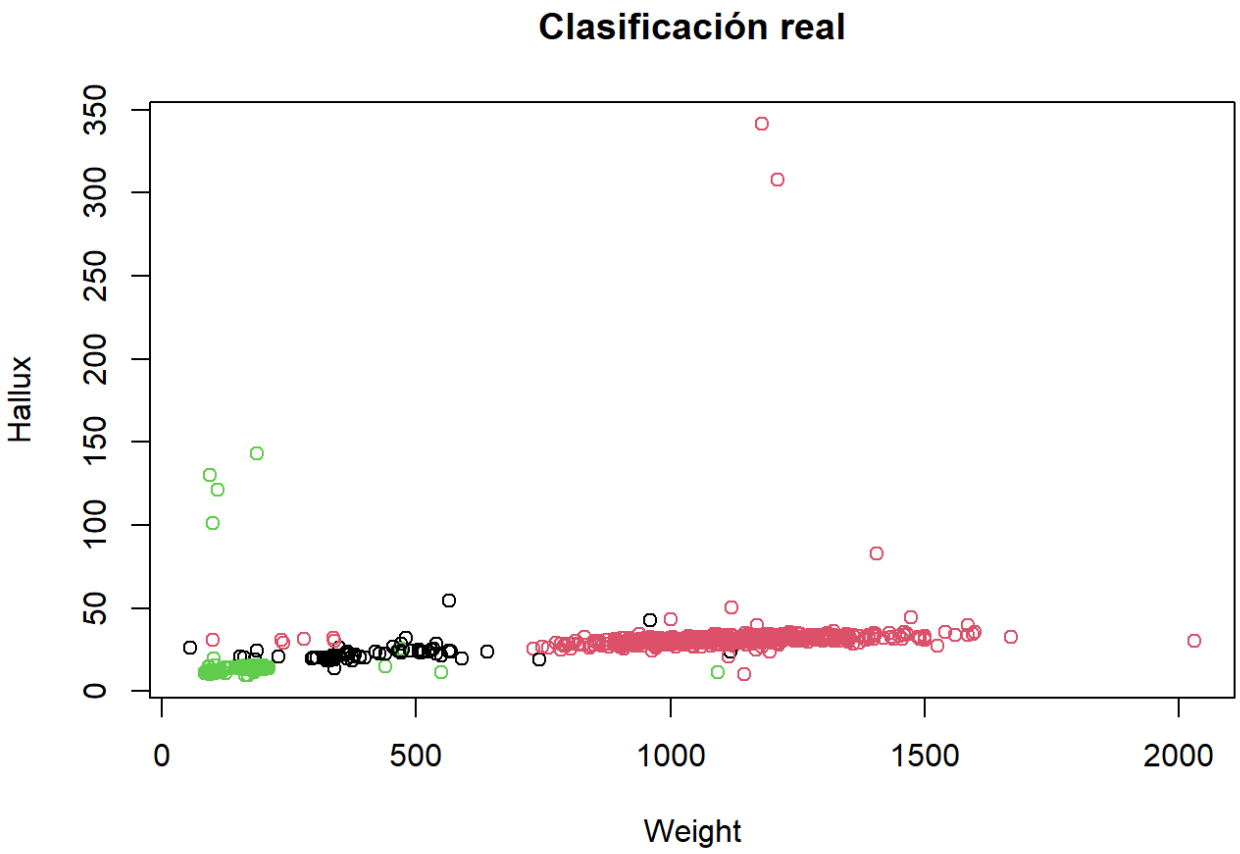

```
plot(Hawks[c(11,12)], col=as.factor(Hawks$Species), main="Clasificación real")
```




```
# Hallux y Weight
plot(x[c(2,4)], col=Hawks3clusters$cluster, main="Clasificación k-means")
```



```
plot(Hawks[c(11,13)], col=as.factor(Hawks$Species), main="Clasificación real")
```



Las conclusiones a las que se llegan a partir de los gráficos anteriores son las siguientes:

- En general las variables son buenos indicadores ya que los tres clústers se pueden observar visualmente separados en la clasificación real.
- A pesar del punto anterior el algoritmo tiene dificultades para identificar uno de los clústers y no devuelve los resultados esperados.

Sería una buena idea añadir diferentes variables del conjunto de datos para comparar los resultados obtenidos y tratar de obtener un algoritmo con mejores resultados.

0.2 Ejercicio 2

0.2.1 Enunciado

Con el juego de datos proporcionado realiza un estudio similar al del ejemplo 2.

- 20%. Se aplican lo algoritmos DBSCAN y OPTICS de forma correcta.
- 25%. Se prueban con diferentes valores de eps.
- 25%. Se obtiene una medida de lo bueno que es el agrupamiento.
- 20%. Se describen e interpretan los diferentes clusters obtenidos.
- 10%. Se presenta el código y es fácilmente reproducible.

0.2.2 Métodos basados en densidad: DBSCAN y OPTICS

Probaremos a continuación con los algoritmos DBSCAN y OPTICS, son métodos de clustering que permiten la generación de grupos no radiales (a diferencia de k-means).

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

A continuación llamamos a la función optics fijando el criterior de vecindad en 10 y dejando el parámetro eps con su valor por defecto.

```
res <- optics(x_scale, minPts = 10)
res
```

```
## OPTICS ordering/clustering for 891 objects.
## Parameters: minPts = 10, eps = 16.7418480888497, eps_cl = NA, xi = NA
## Available fields: order, reachdist, coredist, predecessor, minPts, eps,
##                   eps_cl, xi
```

Observamos el valor de eps que devuelve el algoritmo, eps se trata del límite superior del tamaño de la vecindad épsilon.

A continuación obtenemos el orden de los datos en función de su distancia de alcanzabilidad.

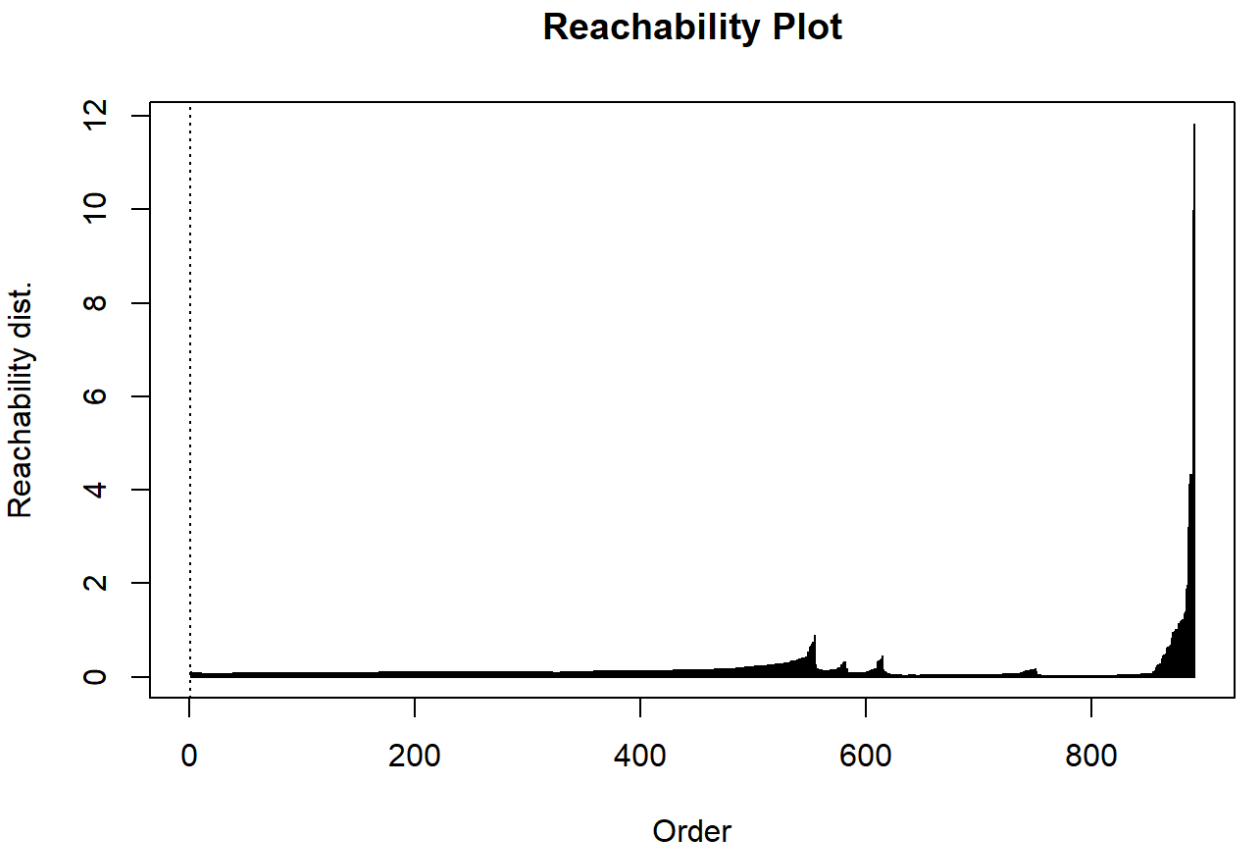
```
res$order
```

```
##   [1]   1 555 380 604 413 773 632 531 414 179  23 808 467 238 111  88 748 473
##  [19] 397 226 216 188 180 114 144 634 771 453 368 308 755 578 129  11 273 197
##  [37] 145 457 275 132 685 828   2 270 160 130 355 845 324 874 816 691 665 494
##  [55] 327 784 516 807 392 855 832 744 459 446 243 746 742 573 492 284 174 149
##  [73]  45  40 758 536 362 108  82 451 344 185 680 655 699 768 563 528  75   6
##  [91] 436  74 126  10 229 881 762 253 701 381  30 860 315 177 539 585 565 749
## [109] 134 545 287 354 314 259 830 663 278 393 164 569 501 599 200 839 724 424
## [127] 410 328 249 703 332 262 580 577  42  28 876 871 443 534  85 766 574 729
## [145] 710 656 512 838 544 851 862 560 556 464 452 420 370 223  17 244 592 266
## [163] 233 176 173 515 601 469 375 152 631 475 191 133  66 510 754 727 633 566
## [181] 423 326 215 150 717 490 846 441 728 542 480  52 178  43 387 105  19 540
## [199] 678  48 286 118 884 852  87 199 131 301 589 586 584 175 693 543 112 345
## [217] 146 491 113 759 890 875 796 550 487 395 251 676 172 168 158 590  67  46
## [235] 148 675 194 202 282 769 154 866 598 537 500 334 811 217 203  24 658 254
## [253] 800 829 575 430 415 367 119 782 642 508  25 716 408 859 850 831 817 689
## [271] 616 503 341 532 378 250 218 700 557 478 432 801 471 535 281 219 625 856
## [289] 761 702 239 181 169 115  77 736 261 247  62  47 255  51 398 283  92   8
## [307] 788  12  90 707 576 525 325 513 374 615 593 292 234 222 357 427 853 840
## [325] 743 667 562 213  55 712 466 285  57 245 581 551 159 121 527 122  84 570
## [343]  20 757 267 257 109 548 520 837 482 753 230 734  76 764 448 231 401 193
## [361] 867 770 869 688 600 201 142 140 594   7 240 101 776   5 660 521 438  29
## [379]  68  97 198 359 591  44 597 272 863 524 455  41 752 715 468 844 653 422
## [397] 379 263 182 377 421 483 449 865 885 135 211 153 774 488 426  16   9 518
## [415] 356 340  33 485 256  89 141 110 100 458 650 496  60 827 277 417 690 295
## [433] 128 289 456 151 265 454 582 382 477 290 252  15 104  72  70 596 498 822
## [451] 329 666 777 751 228 187  34  22 235 842 533 399 391  71 522 649 568 636
## [469]  13 396 877  18 635 847 470 241 523  31 366 372 406 225 248 648  49 886
## [487] 363 460 493 323 157 313 338  65 705 386 883 333  59 465 613  14  36  27
## [505] 288 889 679 781  73 400 264 322 389 484 698 170 237 336  32 538 795 606
## [523] 204 750 335 371 864 274  35 651 442 780 143 779 684  69 887 462 161  83
## [541] 349 236 346 339 337 317 127 186 321 439 587 258 352 388 514 772 793 670
## [559] 664 657 572 561 526 445 302 232   3 390 343 803 403 103  63 686 823 242
## [577] 695 431 428 106 227 429 834 826 814 756 692 627 509 411 361  39 872 835
## [595] 517  54 810 805 792 583 861 741 618  53 291 280 383 214  81 183 626 318
## [613] 726 309 646 719 825 841 879 888 868 880 735 882 731 644 603 709 671 614
## [631] 579 760 628 605 208 196 155 433 404 360 783 812 307 681 623 276 212  99
## [649] 870 672 824 785 730 607 461 305 293 166 165  79 778 733 706 440 820 791
## [667] 786 763 747 722 711 486 878 472 365 608 541 434 806 714 268 819 444 668
## [685] 450 447 718 609 505 499 304 167   4 502 298 425 767 463 504 661 171 418
## [703] 330 787 364 402  26 394 138  78  50 221  93 124 385 673 552 192 530 659
## [721] 595 163 162  94 195 474 220 553 637 677 189 669 798 156 765 210 269 809
## [739] 373 209 331 873 205 804 342 246 350 619 412 224 740 739 674 645 643 836
## [757] 723 641 630 610 571 549 546 821 708 564 437 306 300 639 638 489 435 409
## [775] 511 854 745 721 519 506 407 843 815 833 789 732 720 612 299 296 696 294
## [793] 640 620 416 303 818 617 559 775 611 554 481 405 120 858 647 654 794 857
## [811] 737 622 813 384 376 358 713 588 799 260 694  96 790 558 190 621  38 419
## [829]  80 369 139 137 652 529 495 479  95  64 117 206 125 107 682 547  61 797
## [847] 207 102 849 353 116 697 136 802 320 311 347 310 848 319 123 297 147 312
## [865] 279 316 497 624 738 507 348 891  56 351 683 662 687 629 704 476  21 725
## [883] 271  58 184  37 567 602  98  86  91
```

A continuación observamos el diagrama de alcanzabilidad (reachability plot) en el que se aprecia la distancia de alcanzabilidad de cada punto.

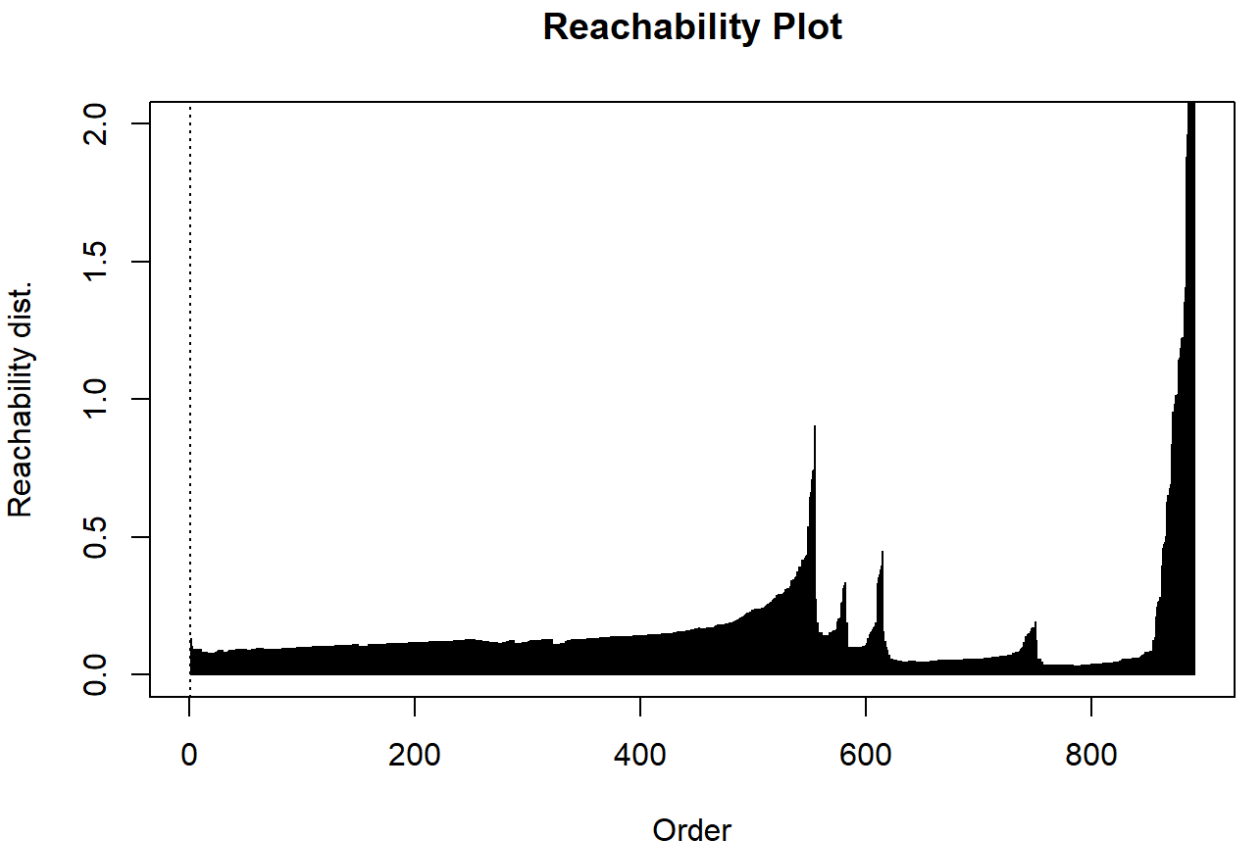
Las cimas indican outliers. Observamos que existen muchos en la última agrupación.

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



A continuación modificamos el límite del eje de las ordenadas para observar mejor el resto de agrupaciones.

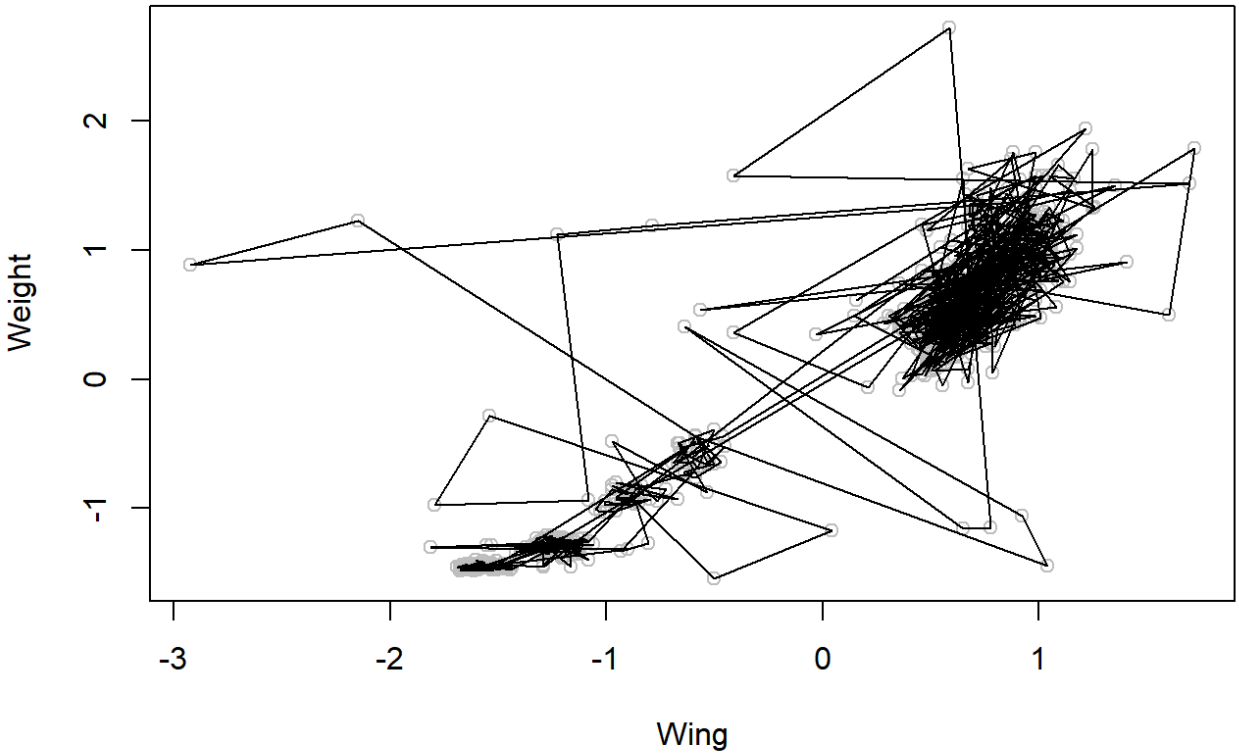
```
plot(res, ylim=c(0,2))
```



Observamos que el algoritmo ha realizado entre 4 y 5 agrupaciones de los datos.

Otra posible representación del diagrama de alcanzabilidad es la siguiente:

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



En este caso visualmente podríamos separar en 3 o 4 agrupaciones y se observa una agrupación bastante más densa que el resto.

A continuación vamos a extraer una agrupación de la ordenación realizada por OPTICS similar a lo que DBSCAN hubiera generado estableciendo el parámetro eps. Se van a estudiar los resultados en caso de modificar el valor de eps_cl. Este valor se utiliza para determinar el número de clústeres, se calcula como un porcentaje de la distancia máxima de alcance en los datos.

En primer lugar probaremos con eps_cl = 0.065

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

```
## OPTICS ordering/clustering for 891 objects.
## Parameters: minPts = 10, eps = 16.7418480888497, eps_cl = 0.065, xi = NA
## The clustering contains 2 cluster(s) and 706 noise points.
##
##   0   1   2
## 706  93  92
##
## Available fields: order, reachdist, coredist, predecessor, minPts, eps,
##                   eps_cl, xi, cluster
```

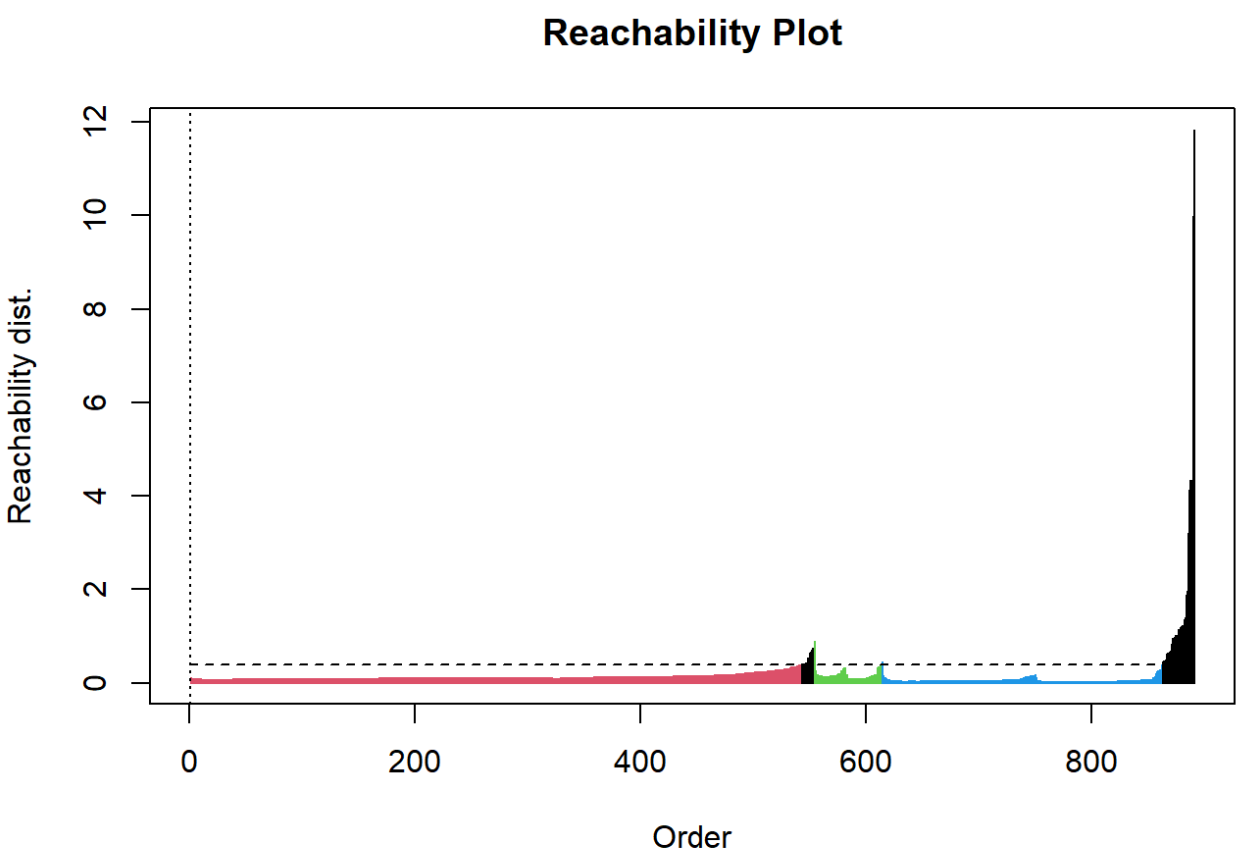
El anterior modelo presenta demasiados “noise points” en comparación con los puntos que aparecen en los dos clústers identificados. Parece que este modelo podría estar obviando un clúster identificando los puntos de éste como “noise points”. Para solucionarlo vamos a probar a disminuir el valor eps_cl.

```
### Extracción de un clustering DBSCAN cortando la alcanzabilidad en el valor eps_cl
resdb <- extractDBSCAN(res, eps_cl = 0.4)
resdb
```

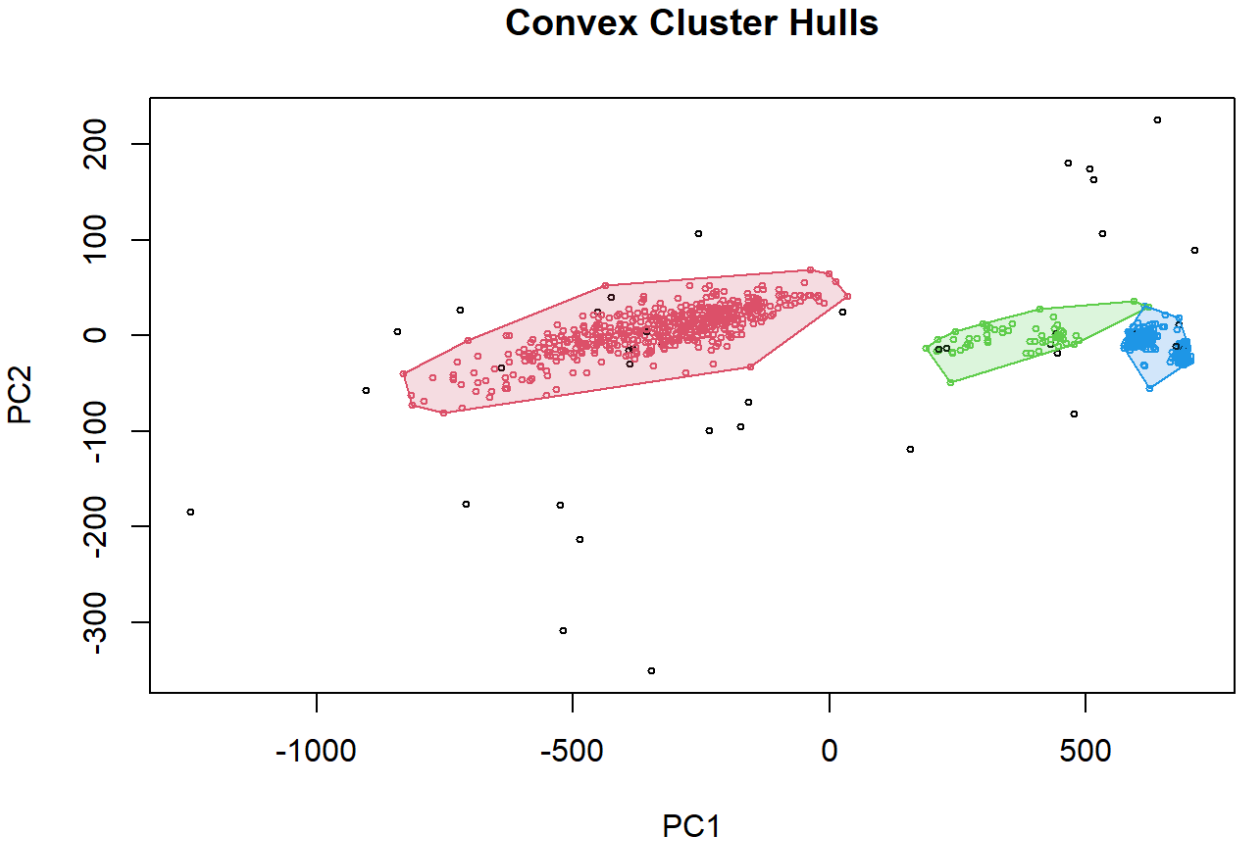
```
## OPTICS ordering/clustering for 891 objects.
## Parameters: minPts = 10, eps = 16.7418480888497, eps_cl = 0.4, xi = NA
## The clustering contains 3 cluster(s) and 41 noise points.
##
##   0   1   2   3
##  41 542  60 248
##
## Available fields: order, reachdist, coredist, predecessor, minPts, eps,
##                   eps_cl, xi, cluster
```

A simple vista parece más razonable, vamos a visualizar el diagrama de alcanzabilidad.

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

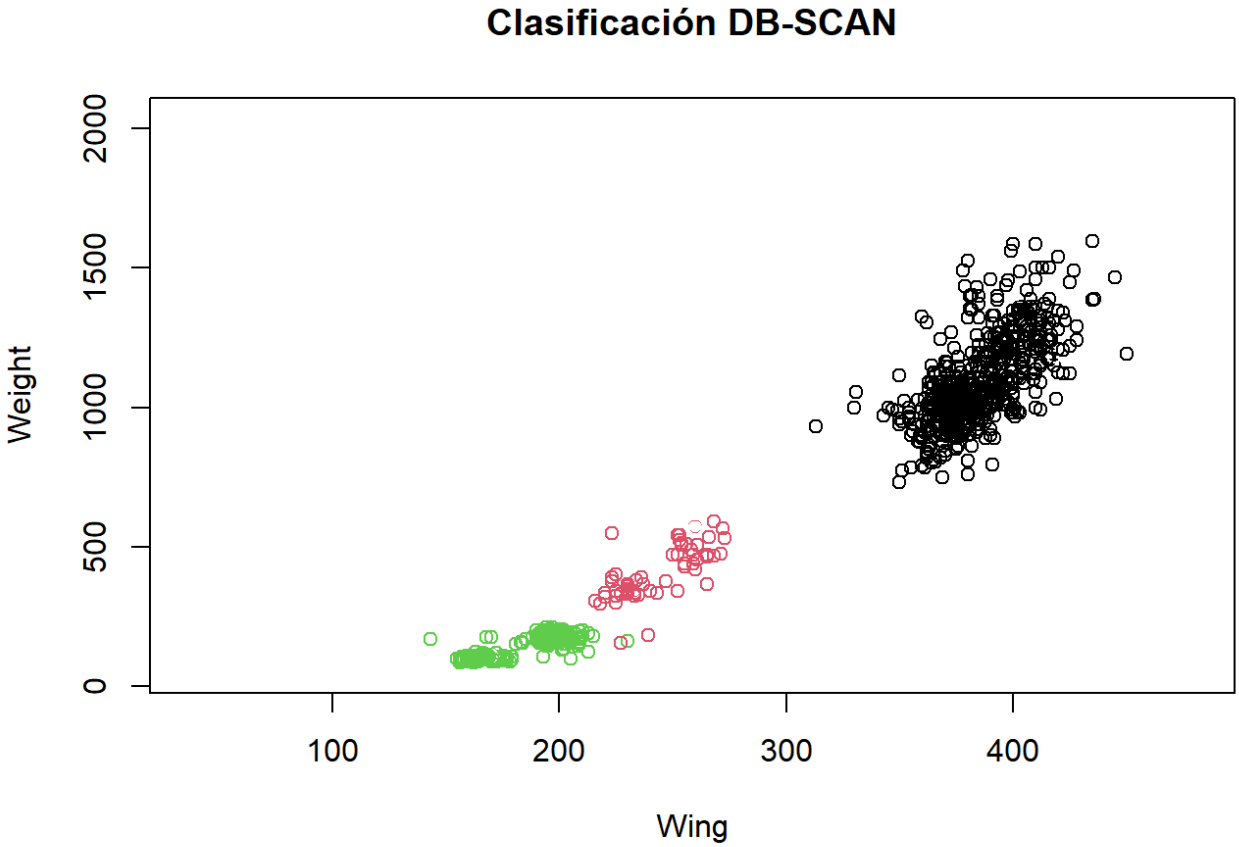



```
hullplot(x, resdb)
```

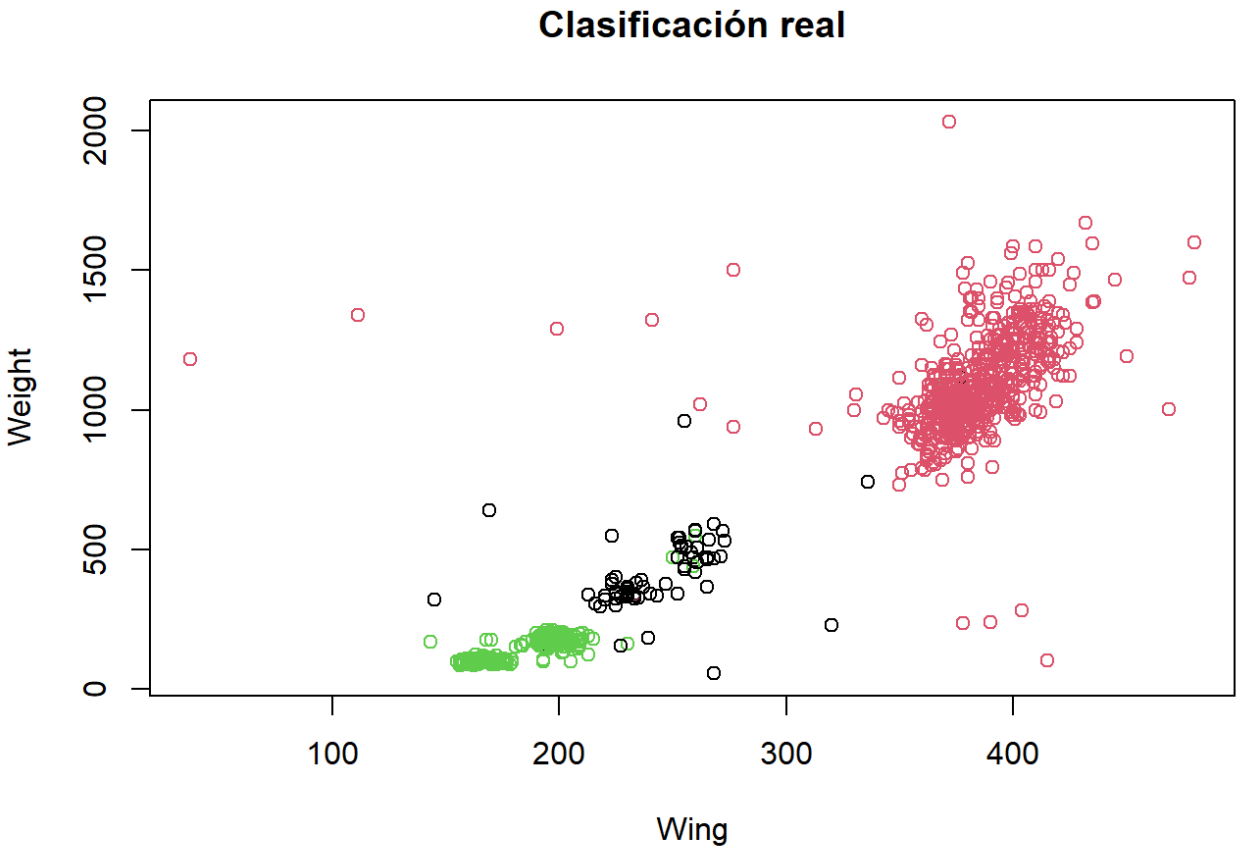


A continuación comprobamos de que manera ha clasificado los datos el algoritmo según la clasificación real.

```
# Wing y Weight
# Gráfico DB-SCAN
plot(x[c(1,2)], col=resdb$cluster, main="Clasificación DB-SCAN")
```




```
# Gráfico de clasificación real
plot(Hawks[c(10,11)], col=as.factor(Hawks$Species), main="Clasificación real")
```



Observamos que ha eliminado los valores extremos y que ha identificado correctamente los tres clústers presentes en el conjunto de los datos.

0.3 Ejercicio 3

0.3.1 Enunciado

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

- 35%. Se comparan los resultados obtenidos en k-means y DBSCAN.
- 35%. Se mencionan pros y contras de ambos algoritmos
- 30%. Se exponen la conclusiones del trabajo

0.3.2 Conclusión

En primer lugar se ha utilizado el algoritmo k-means, que es un algoritmo de clasificación no supervisada, el principal inconveniente de este algoritmo es que se necesita fijar previamente el número de clústers. Lo habitual es no conocerlo por lo que es necesario realizar un estudio previo para determinarlo.

Hemos utilizado diferentes criterios para estudiar el valor ótimo de clústers. Se han realizado dichas pruebas con los valores de k desde 2 hasta 10. Según el método de la silueta media y según el método del codo el valor óptimo de clústers es 3. En cambio el criterio de Calinski-Harabasz ha optado por 6 clústers. Puesto que 3 clústers ha sido la opción óptima en la mayoría de pruebas se ha decidido realizar el modelo con este número.

El modelo de k-means con 3 clústers se ha evaluado visualizando los resultados de las variables de dos en dos. Hemos podido observar que si bien clasifica bastante bien dos de los clústers el tercero no es capaz de identificarlo correctamente.

Los modelos de clustering basados en la densidad más conocidos son DBSCAN y OPTICS. Este tipo de algoritmos se especializan en identificar zonas de alta concentración de observaciones separadas entre sí por zonas con menor densidad de observaciones.

La ventaja de estos algoritmos es que no es necesario identificar el número de clústers previamente. El algoritmo que mejor ha resultado ha sido DBSCAN. El diagrama DBSCAN con eps_cl = 0.4 ha conseguido muy buenos resultados. Se ha observado en la gráfica como éste elimina los valores outliers y ha identificado casi sin error los tres clústers del data set real.

Los motivos por los que ha funcionado mejor DBSCAN que el algoritmo k-means podrían ser lo siguientes:

- DBSCAN es un algoritmo de clustering basado en la densidad que agrupa puntos de datos en regiones densas. En el juego de datos (Hawks) este algoritmo tiende a funcionar mejor por la densidad de sus agrupamientos.
- DBSCAN identifica automáticamente el número de clústers y asigna puntos de datos a éstos en función de su densidad.
- K-means incluye los valores outliers en un clúster incluso si no pertenecen a ninguno. DBSCAN, por otro lado, puede identificarlos.
- K-means es adecuado para identificar clústers esféricos y bien separados, mientras que DBSCAN es más adecuado para identificar clústers de formas arbitrarias y no necesariamente separados.