uc3m | Universidad Carlos III de Madrid

Curso OpenCourseWare

Aprendizaje del Software Estadístico R: un entorno para simulación y computación estadística

Alberto Muñoz García

12. Métodos gráficos para datos multivariantes

Introducción

El análisis multivariante está relacionado con conjuntos de datos que involucran más de una variable para cada unidad de observación. En general tales conjuntos de datos vienen representados por matrices de datos X con n filas y p columnas, donde las filas representan los casos, y las columnas las variables. En cuanto al análisis de tales matrices, unas veces estaremos interesados en estudiar las filas (casos), otras veces las relaciones entre las variables (las columnas) y algunas veces en estudiar simultáneamente ambas cosas.

Graficado sin hacer transformaciones

El primer comando que tenemos disponible para la tarea de explorar datos multivariantes es:

> pairs()

Vamos a ver cómo se puede utilizar para el conjunto de datos iris, que contiene una base de 4 medidas tomadas sobre una muestra de 150 flores ornamentales (lirios). Las medidas tomadas son la longitud y anchura de pétalos y sépalos. En el problema original, se trataba de clasificar las flores en tres especies predeterminadas, utilizando las cuatro medidas mencionadas.

> data(iris) # Cargamos el conjunto de datos iris

> iris[1:4,] # Mostramos 4 registros

Sepal.Length Sepal.Width Petal.Length Petal.Width Species

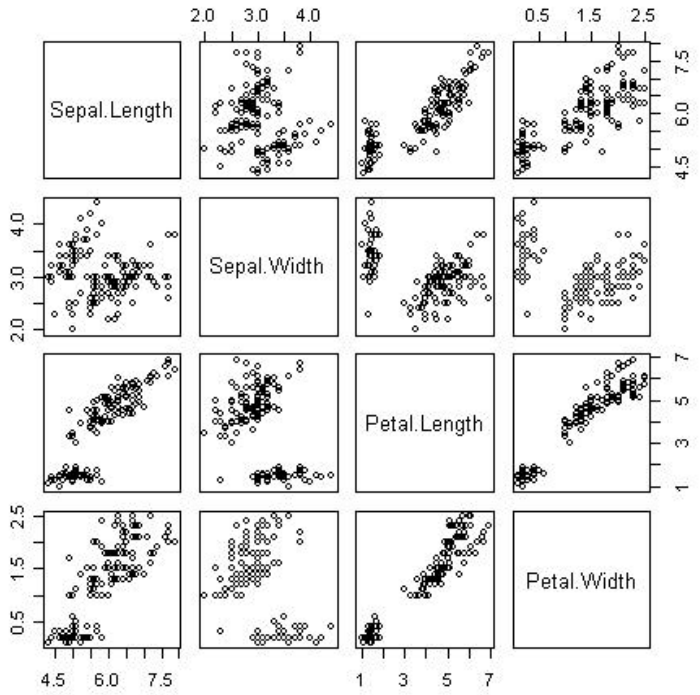
1 5.1 3.5 1.4 0.2 setosa

2 4.9 3.0 1.4 0.2 setosa

3 4.7 3.2 1.3 0.2 setosa

4 4.6 3.1 1.5 0.2 setosa

> pairs(iris[,1:4]) # Mostramos los gráficos de dispersion de las variables



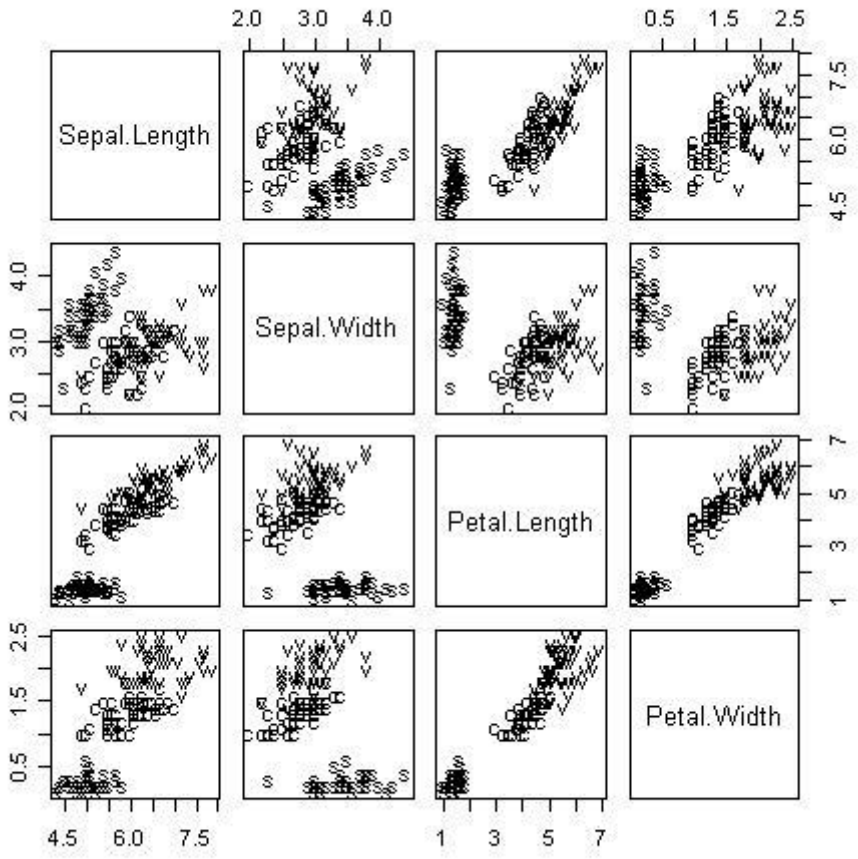
Ahora vamos a dibujar los datos etiquetados por especie. Asi nos daremos cuenta de si las flores de una misma especie aparecen agrupadas con flores de la misma especie. Las flores que aparezcan entre dos grupos corresponderán a flores difíciles de clasificar claramente en uno de los grupos.

Debido a que las etiquetas son bastante largas ("setosa", "virginica", "versicolor"), vamos a sustituirlas por letras, para facilitar la visualización. Usaremos la "s" para setosa, la "v" para virginica, y la "c" para versicolor:

> iris.especies = c(rep("s",50),rep("c",50),rep("v",50))

Para dibujar incluyendo las etiquetas, podemos proceder asi:

> pairs(iris[,1:4],panel=function(x,y,...) text(x,y,iris.especies))



El comando anterior dibujará las coordenadas de los datos, 2 a 2, etiquetando con la clase de cada flor.

Un modo alternativo de proceder es generarnos nosotros mismos gráficos similares. Podríamos hacer (entre otras posibilidades), algo así como:

> par(mfrow=c(1,2))

> plot(iris[,1],iris[,2])

> points(iris[iris.especies=="s",1],iris[iris.especies=="s",2],col=2)

> points(iris[iris.especies=="c",1],iris[iris.especies=="c",2],col=3)

> points(iris[iris.especies=="v",1],iris[iris.especies=="v",2],col=4)

> plot(iris[,1],iris[,3])

> points(iris[iris.especies=="s",1],iris[iris.especies=="s",3],col=2)

> points(iris[iris.especies=="c",1],iris[iris.especies=="c",3],col=3)

> points(iris[iris.especies=="v",1],iris[iris.especies=="v",3],col=4)

La primera orden hace que la pantalla de dibujo quede dividida como una matrix 1 x 2, esto es, una matriz con una fila y dos columnas, por tanto, dos graficos en uno de esa manera.

La segunda orden dibuja en el primer gráfico las coordenadas 1 y 2 de los datos.

La tercera orden y siguientes van repintando los puntos que corresponden a cada especie.

Observad al dar cada orden lo que sucede.

El segundo bloque de ordenes pinta la coordenada 1 frente a la dos y hace lo mismo.

En el caso en que alguien disponga del paquete SPLUS, puede utilizar una orden no implementada aún en R, denominada brush. El modo de empleo sería:

> brush(iris)

Graficado transformando los datos

Cuando los datos tienen más de 3 dimensiones, dibujar las coordenadas 2 a 2 puede ser interesante, pero cuando hay muchas, puede ser más interesante hacer una proyección de los datos usando algún tipo de transformación lineal o no linea, que nos permita resumir en un gráfico de dos dimensiones las características más importantes de los mismos en cuanto a variabilidad, en cuanto a conservación de distancias en la proyección, o en cuanto a la representación gráfica de los clusters hallados en el conjunto de datos.

A continuación pasamos a exponer las diversas técnicas, junto con los comandos en R que las implementan.

Análisis de componentes principales

Esta técnica funciona construyendo automáticamente un conjunto de combinaciones lineales de las variables del conjunto de datos, con la máxima variabilidad posible. Cada componente principal viene determinada por un vector propio de la matriz de covarianzas o correlaciones de los datos, esto es, un vector propio de la matriz $X^T X$. Geométricamente, el primer vector propio de la matriz corresponde a la dirección de máxima variabilidad en el conjunto de datos, que es la dirección sobre la que se proyectará.

Por ejemplo, si tenemos datos sobre una elipse alargada, y hacemos un análisis de componentes principales con una sola componente, la coordenada que representará nuestro conjunto de datos corresponderá a la proyección de los datos sobre el eje principal de la elipse.

Podéis encontrar información más detallada de la teoría en cualquier libro de análisis multivariante, y en la red, podéis acudir a los siguientes links, por ejemplo:

http://cran.r-project.org/doc/vignettes/HSAUR/Ch_principal_components_analysis.pdf(Análisis de componentes principales, Proyecto R).

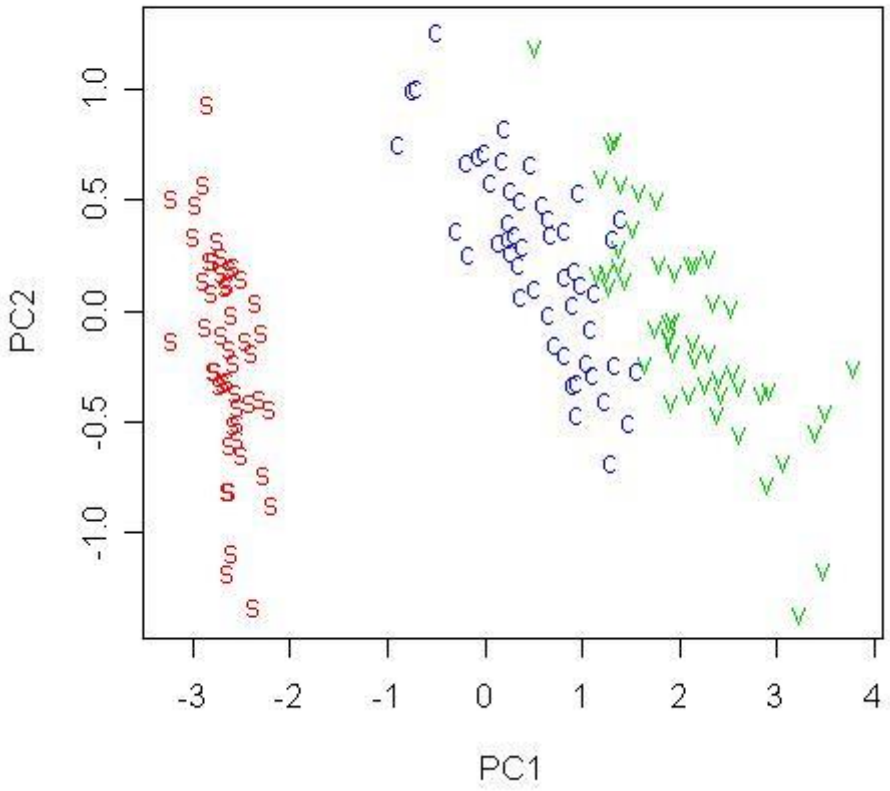
http://csnet.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf(Lindsay I. Smith, Universidad de Otago, Nueva Zelanda).

Para usar componentes principales hay que cargar una librería de las varias que hay con R para análisis multivariante (mva, multiv y otras). Por ejemplo:

```
> library(mva)
> library(help=mva)    # Aqui se ve qué funciones vienen con la libreria
> data(iris)                    # Cargamos los datos de las flores
> iris.pca = prcomp(iris[,1:4])       #  Realizamos el análisis de
componentes principales
> attributes(iris.pca)             # Miramos a ver qué devuelve un tal
análisis
$names
[1] "sdev"     "rotation" "x"

$class
[1] "prcomp"
```

Ahora dibujamos las dos primeras componentes principales y etiquetamos:

```
> plot(iris.pca$x[,1:2],type="n")
> text(iris.pca$x[iris.especies=="s",1:2],col=2,"s")
> text(iris.pca$x[iris.especies=="v",1:2],col=3,"v")
> text(iris.pca$x[iris.especies=="c",1:2],col=4,"c")
```



La estructura iris.pca$x contiene las coordenadas de los puntos, luego al pedir que dibuja las 1:2 estamos pidiendo que dibuje las dos primeras componentes. El type="n" hace que dibuje pero que no se vea, esto es, marca las dos primeras coordenadas. Las demás órdenes dibujan los puntos por clases con un color distinto para cada clase.

Como se puede apreciar en el gráfico, hemos conseguido representar un conjunto de imensión 4 por un conjunto de dimensión 2, y si comparáis con los gráficos ya hechos de los datos iris, puede verse que la estructura se ha respetado bastante fielmente. Puede concluirse que hay 3 grupos, uno bien separado y otros dos más juntos. Y en efecto, así es.

La técnicas de componentes principales podéis emplearla para cualquer conjunto de datos del cual queráis haceros una idea de la estructura. También puede utilizarse para codificar información, en el sentido de sustituir la codificación original de ciertos datos por otra que ocupe menos espacio.

Así por ejemplo, si se dispone de imágenes de fotos (en blanco y negro, representadas por niveles de gris), puede utilizarse la técnica de componentes principales para reducir la dimensión de las imágenes.

Podéis encontrar información más detallada a este respecto en:

http://www.willamette.edu/~gorr/classes/cs449/Unsupervised/pca.html (G. B. Orr, Willamette University, USA).

Para saber cúantas componentes son suficientes para representar fidedignamente un conjunto de datos (por ejemplo, si el conjunto tiene dimensión 10, deberemos usar 2, 3, ¿cuántas componentes?) se miran los valores propios asociados a la descomposición comentada más arriba. Tales valores se guardan en "sdev", de manera que en el ejemplo del iris:

> iris.pca$sdev

[1] 2.0562689 0.4926162 0.2796596 0.1543862

Observamos que los valores se dan de mayor a menor. Tales valors corresponden a variabilidades a lo largo de los ejes de proyección (esto es, las componentes principales). Para hacernos una idea de su importancia relativa, iremos calculando la suma acumulada, dividida por la suma total para que resulte 1:

> cumsum(iris.pca$sdev^2)/sum(iris.pca$sdev^2)

[1] 0.9246187 0.9776852 0.9947878 1.0000000

Como vemos, con 1 componentes se recoge aproximadamente el 92% de la variabilidad de los datos, lo que no es poco a efectos de visualización. También vemos que con 3 componentes ya se explica el 99% de la variabilidad, esto es, una componente no aporta casi nada de información a efectos de explicar la estructura de los datos. La razon de tomar los cuadrados es porque la varianza es el cuadrado de la desviación típica.

Escalado multidimensional

La idea del escalado multidimensional es la misma que la del análisis de componentes principales: Se trata de representar los datos en baja dimensión (usualmente 2), pero en este caso, en vez de utilizar las coordenadas de los datos para llevar a cabo la transformación, se usará la información proporcionada por las distancias entre los datos.

Esta manera de proceder tiene mucho sentido para datos cuyas coordenadas no conocemos , pero de los cuales conocemos las distancias entre ellos. Por ejemplo, en un mapa de carreteras vienen las distancias en kilómetros entre diversas ciudades españolas. Esta técnica permite reconstruir las coordenadas de los datos, conociendo las distancias entre los mismos.

Podéis consultar ejemplos y la teoría más detallada en:

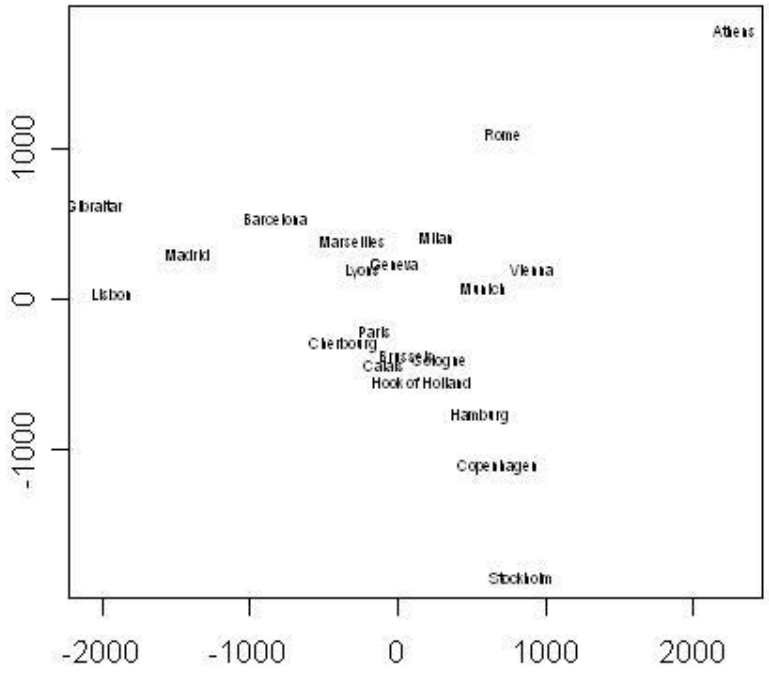
http://www.analytictech.com/networks/mds.htm (Analytic Technologies, USA)

Otro caso en el que el escalado multidimensional resulta útil se da cuando la información viene dada por matrices de preferencias. Este es el caso cuando se pide a un grupo de alumnos que asignen preferencias al resto de sus compañeros.

Como primer ejemplo, consideraremos el que viene con la función cmdscale de R (libreria mva):

```
> data(eurodist)    # Cargamos los datos en memoria
> attributes(eurodist)
$Size
[1] 21

$Labels
[1] "Athens"        "Barcelona"     "Brussels"      "Calais"
[5] "Cherbourg"     "Cologne"       "Copenhagen"    "Geneva"
[9] "Gibralta"      "Hamburg"       "Hook of Holland" "Lisbon"
[13] "Lyons"        "Madrid"        "Marseilles"    "Milan"
[17] "Munich"       "Paris"         "Rome"          "Stockholm"
[21] "Vienna"

$class
[1] "dist"

> eurodist.mds = cmdscale(eurodist)

> x = eurodist.mds[,1]
> y = eurodist.mds[,2]
> plot(x, y, type="n", xlab="", ylab="")

> eurodist2 = as.matrix(eurodist)
```

> text(x, y, dimnames(eurodist2)[[1]], cex=0.5)



Así pues, eurodist es un objeto que guarda los nombres de las distancias, la matriz de distancias entre las ciudades, así como el número de objetos, que se podría recuperar escribiendo:

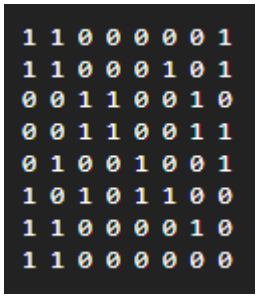
> attr(eurodist,"Size")

[1] 21

El proceso de escalado se realiza con la función cmdscale y después se dibuja el mapa de las ciudades con sus nombres. Podéis repetir el proceso con algunas ciudades españolas. El mapa que obtendréis puede aparecer rotado o invertido, dado que el método garantiza la conservación de las distancias, pero eso no garantiza que la orientación en el plano sea la correcta.

A continuación vamos a dibujar un mapa de preferencias entre un grupo de 8 personas. Cada persona asigna 1 a las personas de su preferencia. Dicho mapa reflejará grupitos de amigos. Observad que en la diagonal hay unos, asumiendo que uno es amigo de sí mismo.

Los datos podéis encontrarlos aquí:



La matriz de datos presenta el siguiente aspecto:

paco pepe lola juan carlos luis ana pedro

paco 1 1 0 0 0 0 0 1

pepe 1 1 0 0 0 1 0 1

lola 0 0 1 1 0 0 1 0

juan 0 0 1 1 0 0 1 1

carlos 0 1 0 0 1 0 0 1

luis 1 0 1 0 1 1 0 0

ana 1 1 0 0 0 0 1 0

pedro 1 1 0 0 0 0 0 1

Para leer dicha matriz de un archivo, se puede leer directamente, como hago yo a continuación, o de otra forma que prefiráis. El directorio que pongo yo en la orden scan es el mío. En vuestro caso, podrá variar.

> amigos.txt = scan("c:\\amigos.txt")

Read 64 items

> amigos.txt = structure(amigos.txt,dim=c(8,8),byrow=T)

> amigos.txt = t(amigos.txt)

> amigos.txt

[,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]

[1,] 1 1 0 0 0 0 0 1

[2,] 1 1 0 0 0 1 0 1

[3,] 0 0 1 1 0 0 1 0

[4,] 0 0 1 1 0 0 1 1

[5,] 0 1 0 0 1 0 0 1

[6,] 1 0 1 0 1 1 0 0

[7,] 1 1 0 0 0 0 1 0

[8,] 1 1 0 0 0 0 0 1

attr(,"byrow")

[1] TRUE

> rownames(amigos.txt)<-c("paco","pepe","lola","juan","carlos","luis","ana","pedro")

> colnames(amigos.txt)<-c("paco","pepe","lola","juan","carlos","luis","ana","pedro")

> amigos.txt

> plot(amigos.mds[,1],amigos.mds[,2],type="n")

> text(amigos.mds[,1],amigos.mds[,2],labels=rownames(amigos.txt))

El comando rownames asigna nombres a las filas, y colnames hace lo propio con las columnas. El gráfico final os mostrará una imagen bidimensional de las preferencias de esas personas

Análisis de cluster

El análisis de cluster, denominado en español a veces análisis de conglomerados, es una técnica no supervisada para descubrir la estructura de datos de un conjunto. Entra en el apartado de técnicas gráficas para datos multivariantes por el siguiente motivo: Cuando unos datos son de alta dimensión, es bastante dificil descubrir cuántos grupos hay en el conjunto de datos, y qué dato pertenece a cada grupo, en su caso. El análisis de cluster engloba una serie de técnicas para dividir el conjunto de datos en una serie de grupos, y frecuentemente se apoya en métodos de visualización propios para la visualización de tales grupos.

La información que se usa para llevar a cabo la mayoría de los procedimientos de análisis de cluster es la proporcionada por las distnacias entre los datos, al igual que en el caso del escalado.

Aunque hay varias maneras de llevar a cabo la construcción de los grupos, hay dos grandes grupos de técnicas: las técnicas jerárquicas y las técnicas no jerárquicas.

Las técnicas jerárquicas proceden por niveles a la hora de construir los grupos. En un caso se procede de abajo hacia arriba, considerando tantos grupos como datos hay en el conjunto, y agrupando sucesivamente hasta que solo queda un grupo. La visualización del procedimiento utilizando una herramienta gráfica denominada dendrograma, ayudará a descubrir la estructura de grupos en los datos. Para ir agrupando los datos, primero se buscan los datos

más cercanos entre sí. En el segundo nivel se agrupan los grupos más cercanos entre si, y asi sucesivamente. El otro caso frecuente es partir de un sólo grupo e ir partiendolo en grupos más pequeños hasta llegar al nivel de los datos individuales. Para todas estas técnicas jerárquicas hay librerías en R. Aquí utilizaremos la libreria mva, y dentro de ella la rutina más popular, hclust.

Para ilustrar el funcionamiento, vamos a utilizar una base de animales descritos por algunas características.

Los datos están en:

animal.txt

animal.des

anima2.des

Leemos los datos:

> animal = scan("c:\\animal.txt")

Read 208 items

> animal = structure(animal,dim=c(13,16),byrow=T)

Hasta aquí tenemos la matrix animal como una matriz de 16 animales por 13 caracteristicas. Ahora leemos los nombres de los animales y de las caracteristicas:

animal.nombres=

c("paloma","gallina","pato","ganso","lechuza","halcon","aguila","zorro","perro","lobo","gato", "tigre","leon","caballo","cebra","vaca")

> animal.nombres

[1] "paloma" "gallina" "pato" "ganso" "lechuza" "halcon" "aguila" "zorro"

[9] "perro" "lobo" "gato" "tigre" "leon" "caballo" "cebra" "vaca"

animal.carac=

c("peque","mediano","grande","bipedo","cuadrupedo","pelo","pezunas","melena","plumas","caza","corre","vuela","nada")

Asignamos nombres a filas y columnas:

> dimnames(animal) = list(animal.nombres,animal.carac)

Y por último, echamos un vistazo a una parte de la matriz:

> animal[1:5,1:10]

peque mediano grande bipedo cuadrupedo pelo pezu¤as melena plumas caza

paloma 1 0 0 1 0 0 0 0 1 0

gallina 1 0 0 1 0 0 0 0 1 0

pato 1 0 0 1 0 0 0 0 1 0

ganso 1 0 0 1 0 0 0 0 1 0

lechuza 1 0 0 1 0 0 0 0 1 1

Podéis mirar toda la matriz, y ver que las descripciones son correctas, aunque algo simples, por supuesto.

Nuestro objetivo en un análisis de cluster sería visualizar de alguna manera la estructura de los datos, y descubrir qué grupos de animales pueden existir.

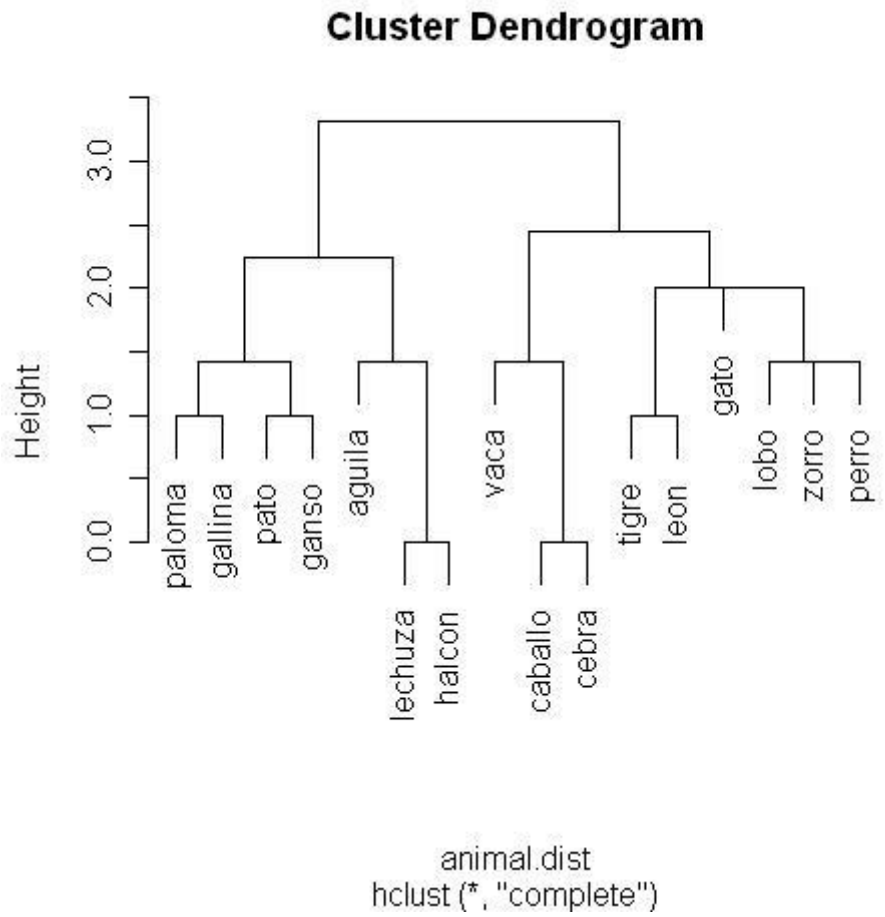
Para ello, calcularemos la matriz de distancias entre los animales, descritos por sus características, realizaremos el análisis de cluster sobre los grupos usando la función hclust, y después visualizaremos los grupos utilizando el dendrograma:

> library(mva)

> animal.dist = dist(animal)

> animal.hclust = hclust(animal.dist)

> plot(animal.hclust)



Como podéis ver al repetir estos comandos, salen dos grandes grupos, uno con las aves, y otro con los mamiferos. Dentro de las aves, veréis un grupo formado por las rapaces y otro por las demás, y dentro de los mamíferos, una distinción entre mamíferos grandes y menos grandes, distinguiendo los cazadores de los que no lo son. ¿A que queda bonito?

Esta técnica aplicada a datos menos evidentes, puede ayudarnos a descubrir grupos interesantes en los datos bajo estudio.