Sesion 5

Mariana Vélez

2022-05-02

Análisis multivariados

A veces hay muchas variables en un sistema y necesitamos responder preguntas como: ¿Cuáles son las variables más importantes en un set de datos? ¿Qué variables son discriminantes? ¿Qué tan parecidas son las muestras entre sí? Y para eso usamos técnicas multivariadas como análisis de ordenación, análisis de clúster o análisis de discriminantes.

Técnicas de ordenacion

Análisis de Componentes Principales (PCA)

Principal Component Analysis (PCA) es un método estadístico que permite simplificar la complejidad de espacios muestrales con muchas dimensiones a la vez que conserva su información. El método de PCA permite por lo tanto "condensar" la información aportada por múltiples variables en solo unas pocas componentes. Una de las aplicaciones de PCA es la reducción de dimensionalidad (variables), perdiendo la menor cantidad de información (varianza) posible: cuando contamos con un gran número de variables cuantitativas posiblemente correlacionadas (indicativo de existencia de información redundante), PCA permite reducirlas a un número menor de variables transformadas (componentes principales) que expliquen gran parte de la variabilidad en los datos. Cada dimensión o componente principal generada por PCA será una combinación lineal de las variables originales, y serán además independientes o no correlacionadas entre sí

El PCA permite ver como se relacionan las variables entre sí a través de la correlación, permite ver cuanto porcentaje de la variación está relacionado con qué variables y como se ubican las muestras entre todo ese gradiente de diferentes combinaciones de las variables. En R, el paquete ade4 trae funciones para hacer PCA

Supuestos:

- Normalidad
- Linealidad
- Muestreo aleatorio

```
library(ade4)
```

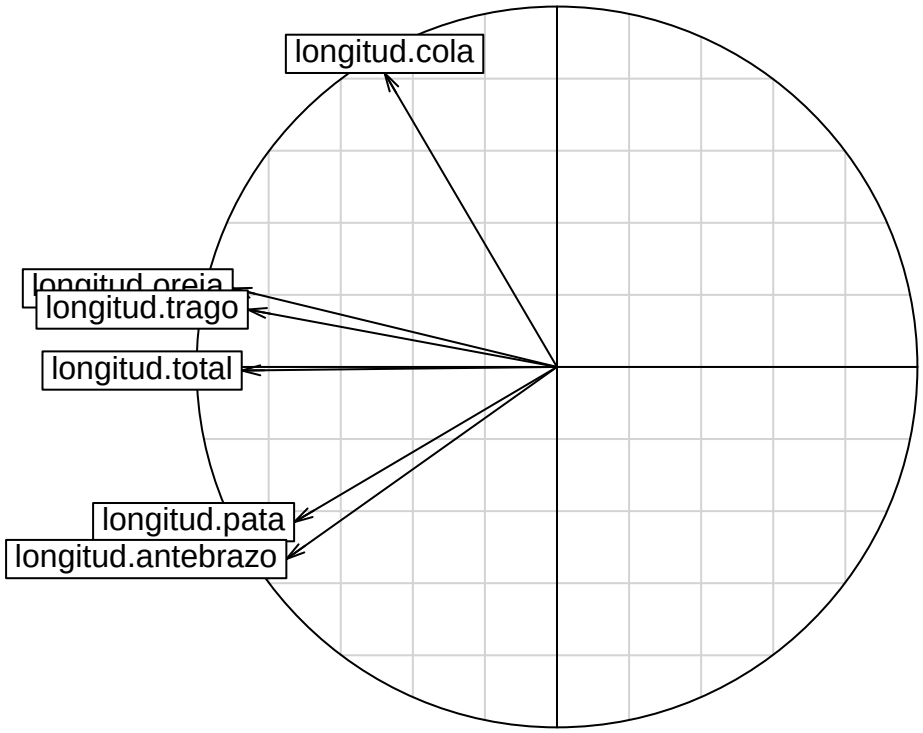
```
## Warning: package 'ade4' was built under R version 4.1.3
```

```
bats<-read.delim("Murcis.txt")
bats<-bats[,c(1,10,13:18)]
bats<-na.omit(bats)
pca<-dudi.pca(bats[,3:8],center = TRUE,scale = TRUE,scannf = FALSE)
summary(pca)
```

```
## Class: pca dudi
## Call: dudi.pca(df = bats[, 3:8], center = TRUE, scale = TRUE, scannf = FALSE)
##
## Total inertia: 6
##
```

```
## Eigenvalues:
##     Ax1     Ax2     Ax3     Ax4     Ax5
##  3.6356  1.2080  0.4114  0.3746  0.2292
##
## Projected inertia (%):
##     Ax1     Ax2     Ax3     Ax4     Ax5
##  60.594  20.133   6.857   6.243   3.819
##
## Cumulative projected inertia (%):
##     Ax1   Ax1:2   Ax1:3   Ax1:4   Ax1:5
##   60.59   80.73   87.58   93.83   97.65
##
## (Only 5 dimensions (out of 6) are shown)
```

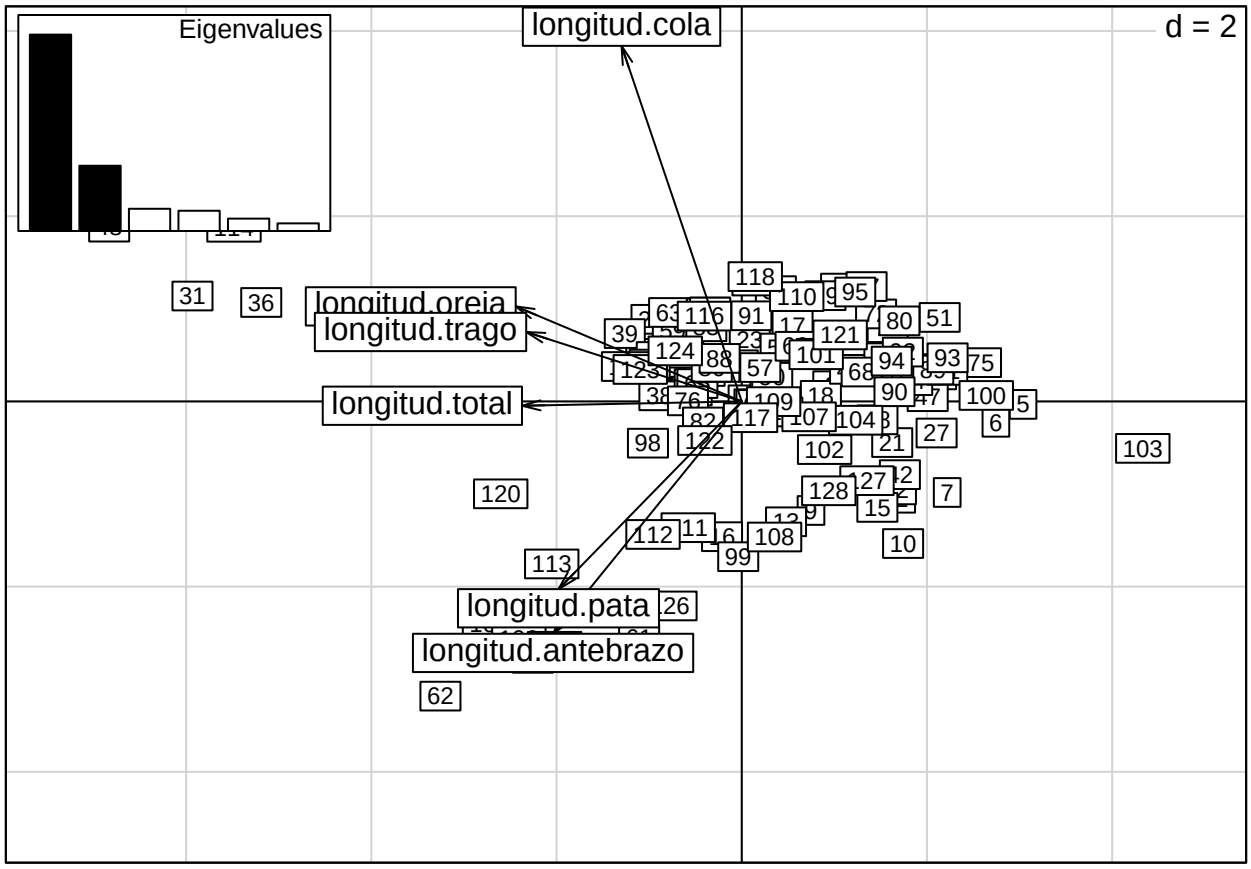
```
s.corcircle(pca$co)
```




```
pca$co
```

```
##                       Comp1        Comp2
## longitud.total     -0.8747727 -0.009944179
## longitud.cola      -0.4783911  0.815703229
## longitud.pata      -0.7293718 -0.430631833
## longitud.oreja     -0.8998136  0.218038754
## longitud.trago     -0.8578624  0.159313388
## longitud.antebrazo -0.7509761 -0.533037893
```

```
biplot(pca)
```



Center es un argumento de tipo lógico o numérico que indica cómo se está centrando la variable, si center es TRUE se está centrando a partir de la media, si center es FALSE no se está centrando y si center es numérico es porque se está dando un vector con el número de columnas del data frame y da el descentramiento

Scale es un argumento lógico que indica si las columnas están siendo puestas en la misma escala de manera que sean comparables entre sí.

Análisis de Correspondencias Canónicas

El Análisis de Correspondencia Canónica (CCA) sirve para analizar una tabla de contingencia (habitualmente con ubicaciones como filas y especies en columnas) teniendo en cuenta la información proporcionada por un conjunto de variables explicativas contenidas en una segunda tabla y medidas en las mismas ubicaciones. El Análisis de Correspondencia Canónica (CCA) permite relacionar la abundancia de especies con las variables del entorno.}

Supuestos:

- Hay un gradiente ambiental
- La distribución de las especies es unimodal

```
library(vegan)
```

```
## Warning: package 'vegan' was built under R version 4.1.3
```

```
## Loading required package: permute
```

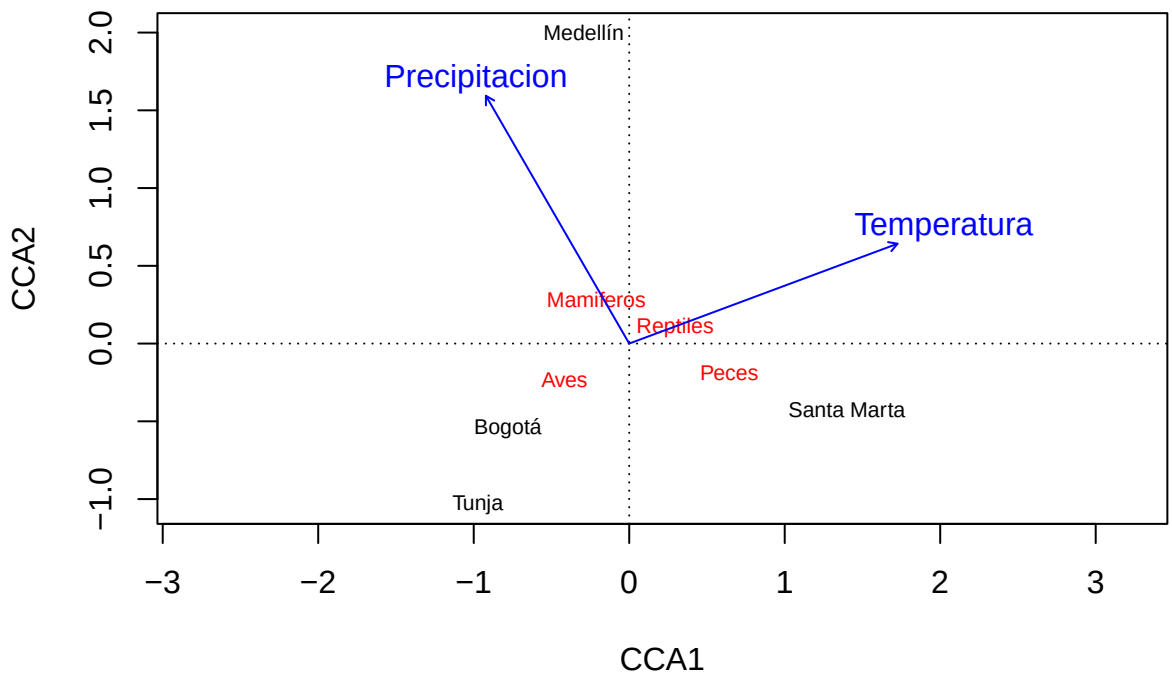
```
## Loading required package: lattice
```

```
## This is vegan 2.6-2
```

```r
ciudades<-c("Medellín","Bogotá","Santa Marta","Tunja")
mamiferos<-c(45,23,20,21)
reptiles<-c(25,10,45,19)
peces<-c(9,10,45,7)
aves<-c(20,34,21,50)
temp<-c(25,17,30,18)
precip<-c(2958,1091,977,2026)


datos<-matrix(data = c(mamiferos,reptiles,peces,aves,temp,precip),nrow = 4,ncol = 6)
row.names(datos)<-ciudades
colnames(datos)<-c("Mamiferos","Reptiles","Peces","Aves","Temperatura","Precipitacion")

cca<-cca(datos[,1:4],datos[,5:6])
plot(cca)
```



Técnicas de clasificación

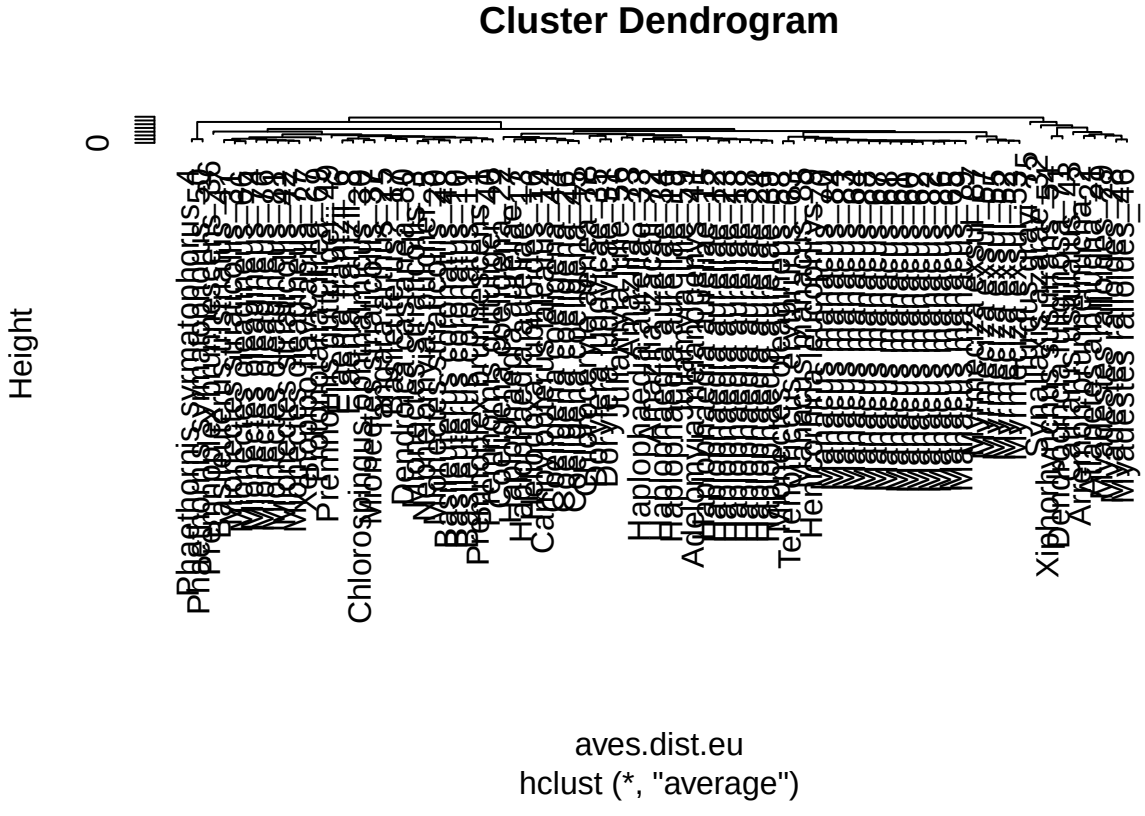
Agrupamiento jerárquico

El agrupamiento es básicamente una técnica que agrupa puntos de datos similares de modo que los puntos en el mismo grupo son más similares entre sí que los puntos en los otros grupos. El grupo de puntos de datos similares se llama Cluster. En el caso del agrupamiento jerarquico, los puntos se van clasificando en grupos pequeños que pertenecen a grupos más grandes.

```
aves<-read.table("aves.txt",header = TRUE,sep="\t")
aves.c <- na.omit(aves[,c(1,3,10:11,15,19:29)])
row.names(aves.c)<-paste(aves.c$Especie, 1:88, sep="_")
aves.dist.eu<-vegdist(aves.c[,6:16], method="euclidean")
as.matrix(aves.dist.eu)[1:6,1:3]
```

```
##                              Premnoplex brunnescens_1 Arremon brunneinucha_2
## Premnoplex brunnescens_1                      0.00000                44.61424
## Arremon brunneinucha_2                       44.61424                 0.00000
## Mionectes striaticollis_3                    10.33296                45.52626
## Phaethornis syrmatophorus_4                  48.26251                65.00123
## Nephelomyias pulcher_5                       14.53513                46.26230
## Elaenia frantzii_6                           26.80336                40.61539
##                              Mionectes striaticollis_3
## Premnoplex brunnescens_1                     10.332957
## Arremon brunneinucha_2                       45.526256
## Mionectes striaticollis_3                     0.000000
## Phaethornis syrmatophorus_4                  52.922018
## Nephelomyias pulcher_5                        4.707441
## Elaenia frantzii_6                           19.069609
```

```
aves.cluster<- hclust(aves.dist.eu, method="average")
plot(aves.cluster)
```



Análisis de función discriminante

El análisis de función discriminante (DFA) es una técnica de "clasificación", introducida por Fisher. DFA se usa cuando tenemos observaciones de grupos predeterminados con dos o más variables de respuesta registradas para cada observación. DFA genera una combinación lineal de variables que maximiza la probabilidad de asignar correctamente las observaciones a sus grupos predeterminados y también se puede utilizar para clasificar nuevas observaciones en uno de los grupos. También podríamos desear tener alguna medida de la probabilidad de éxito de nuestra clasificación.

Supuestos: - No multicolinearidad - Normalidad - Homocedasticidad

En R se puede hacer usando el paquete MASS, la funcion lda() data

```
library(MASS)
data("iris")
dis <- lda(Species~Sepal.Length+Petal.Length+Sepal.Width+Petal.Width, data=iris,
prior=c(1/3,1/3,1/3))
dis
```

```
## Call:
## lda(Species ~ Sepal.Length + Petal.Length + Sepal.Width + Petal.Width,
##     data = iris, prior = c(1/3, 1/3, 1/3))
##
## Prior probabilities of groups:
##     setosa versicolor  virginica
##  0.3333333  0.3333333  0.3333333
##
## Group means:
##            Sepal.Length Petal.Length Sepal.Width Petal.Width
## setosa            5.006        1.462       3.428       0.246
## versicolor        5.936        4.260       2.770       1.326
## virginica         6.588        5.552       2.974       2.026
##
## Coefficients of linear discriminants:
##                     LD1         LD2
## Sepal.Length  0.8293776  0.02410215
## Petal.Length -2.2012117 -0.93192121
## Sepal.Width   1.5344731  2.16452123
## Petal.Width  -2.8104603  2.83918785
##
## Proportion of trace:
##    LD1    LD2
## 0.9912 0.0088
```

```
dato1<-c(5,1.5,3.5,0.3)
dato2<-c(6.5,5.5,3,2)
datosnuevos<-rbind(dato1,dato2)
datosnuevos<-as.data.frame(datosnuevos)
colnames(datosnuevos)<-c("Sepal.Length","Petal.Length", "Sepal.Width", "Petal.Width")
predict(dis,newdata = datosnuevos)
```

```
## $class
## [1] setosa    virginica
## Levels: setosa versicolor virginica
##
## $posterior
##             setosa   versicolor    virginica
```

```
## dato1 1.000000e+00 8.060220e-20 6.884999e-39
## dato2 8.737463e-39 3.270596e-04 9.996729e-01
##
## $x
##             LD1       LD2
## dato1  7.477695 0.4887371
## dato2 -5.628104 0.5415642
```