Ecología de comunidades en R- clase 3

Ph D.Stephanie Hereira-Pacheco
CTBC
UATx

15 - 03 - 2022

Análisis multivariados en datos ecológicos

En este capítulo vamos a explorar las diferentes técnicas de análisis multivariado para explorar nuestros datos ecológicos o biológicos.

- Matrices de distancia
- Análisis de cluster
- Métodos de ordenación: PCA, PCoA, CCA, NMDS
- perMANOVA

Matrices de (dis)similitud

Como lo vimos en el módulo teórico las matrices de dis(similitud) son de gran utilidad para muchos análisis en ecología de comunidades. Así que primero veamos como obtener nuestras matrices.
R posee una función predeterminada para obtenerla: **dist()**. Inicialmente llamaré al paquete vegan para cargar la data de **varespec**, esta data nos presenta 44 especies de pastos de liquenes (lichen pastures), los nombres de las columnas son nombres formados de los nombres cientificos de estas especies y los valores que nos presenta son de cobertura estimada:

```
library(vegan)
data("varespec")
varespec[1:6, 1:4]
```

```
##    Callvulg Empenigr Rhodtome Vaccmyrt
## 18     0.55    11.13     0.00     0.00
## 15     0.67     0.17     0.00     0.35
## 24     0.10     1.55     0.00     0.00
## 27     0.00    15.13     2.42     5.92
## 23     0.00    12.68     0.00     0.00
## 19     0.00     8.92     0.00     2.42
```

Matrices de (dis)similitud

Ahora sí apliquemos la función **dist()** para obtener una matriz euclidiana:

```
eucl_dist<- dist(varespec, method = "euclidean")
```

Matrices de (dis)similitud

```
eucl_dist
```

```
##           18       15       24       27       23       19       22
## 15  40.37368
## 24  46.27477 28.35874
## 27  59.87344 30.49724 39.77855
## 23  24.54328 26.60815 33.21226 41.78866
## 19  34.27126 31.61316 36.60304 44.23517 27.04775
## 22  49.33512 30.86003 45.82774 48.59480 38.64358 43.91427
## 16  35.98314 27.72034 40.70025 48.96734 31.22161 35.84254 19.69141
## 28  76.28558 42.45458 50.01474 25.42484 59.74367 58.06714 60.04516
## 13  29.72944 47.05861 53.53636 67.42766 42.33547 40.18469 54.19887
## 14  35.94866 38.79924 42.90444 61.39094 35.65736 38.80321 41.17716
## 20  22.06805 27.36560 31.18323 46.93335 14.46833 25.17124 39.18976
## 25  40.47882 21.53302 20.39499 37.83977 25.72712 30.43738 32.24576
## 7   28.69034 57.88555 64.55017 77.62356 50.38028 56.44443 67.09804
## 5   43.55582 65.97081 70.81557 83.18682 60.33552 64.61900 74.62079
## 6   27.15780 57.20421 63.97989 77.36307 49.50250 51.52029 66.54232
## 3   56.62383 71.76913 74.05306 84.19067 65.99354 47.35542 76.83357
## 4   33.67680 50.83691 54.76934 69.60520 45.05405 31.23740 58.31114
## 2   75.42232 87.14308 87.93849 94.99337 80.91850 60.42866 90.50360
## 9   85.79913 92.53898 92.84378 99.39486 86.92633 65.93488 95.29205
```

Matrices de (dis)similitud

Dentro de los métodos que podemos escoger con esta función encontramos: “euclidean”, “maximum”, “manhattan”, “canberra”, “binary” o “minkowski”. Como vemos la mayoria de estas distancias no son tan aplicadas en ecologia sino en otros tipos de datos como los económicos.

Matrices de (dis)similitud

El paquete **vegan()** tiene también su función para obtener las matrices de distancia y es **vegdist()**, siguiendo el ejemplo anterior:

```
eucl_dist<- vegdist(varespec, method = "euclidean")
```

Matrices de (dis)similitud

```
eucl_dist
```

```
##           18       15       24       27       23       19       22
## 15  40.37368
## 24  46.27477 28.35874
## 27  59.87344 30.49724 39.77855
## 23  24.54328 26.60815 33.21226 41.78866
## 19  34.27126 31.61316 36.60304 44.23517 27.04775
## 22  49.33512 30.86003 45.82774 48.59480 38.64358 43.91427
## 16  35.98314 27.72034 40.70025 48.96734 31.22161 35.84254 19.69141
## 28  76.28558 42.45458 50.01474 25.42484 59.74367 58.06714 60.04516
## 13  29.72944 47.05861 53.53636 67.42766 42.33547 40.18469 54.19887
## 14  35.94866 38.79924 42.90444 61.39094 35.65736 38.80321 41.17716
## 20  22.06805 27.36560 31.18323 46.93335 14.46833 25.17124 39.18976
## 25  40.47882 21.53302 20.39499 37.83977 25.72712 30.43738 32.24576
## 7   28.69034 57.88555 64.55017 77.62356 50.38028 56.44443 67.09804
## 5   43.55582 65.97081 70.81557 83.18682 60.33552 64.61900 74.62079
## 6   27.15780 57.20421 63.97989 77.36307 49.50250 51.52029 66.54232
## 3   56.62383 71.76913 74.05306 84.19067 65.99354 47.35542 76.83357
## 4   33.67680 50.83691 54.76934 69.60520 45.05405 31.23740 58.31114
## 2   75.42232 87.14308 87.93849 94.99337 80.91850 60.42866 90.50360
## 9   85.79913 92.53898 92.84378 99.39486 86.92633 65.93488 95.29205
```

Matrices de (dis)similitud

La ventaja de esta función es que posee métodos como “manhattan”,“canberra”, “clark”, “bray”, “kulczynski”, “jaccard”, “gower”, “altGower”, “morisita”, “horn”, “mountford”, “raup”, “binomial”, “chao”, “cao”, “mahalanobis”, “chisq” ó “chord” que suelen ser distancias más conocidas y aplicables en ecología de comunidades.

Análisis de agrupamiento

El análisis de clusters o agrupamiento se aplica sobre una matriz de distancia previamente obtenida. Como vimos en el módulo teórico hay varios métodos para realizar este clustering, tales como: simple (single), completo (complete) y promedio (average).

La función **hclust()** de R nos permite explorar todas estas formas y otras más como ward.D2, mendian y centroid, comúnmente usados en análisis de datos biológicos.

Análisis de agrupamiento

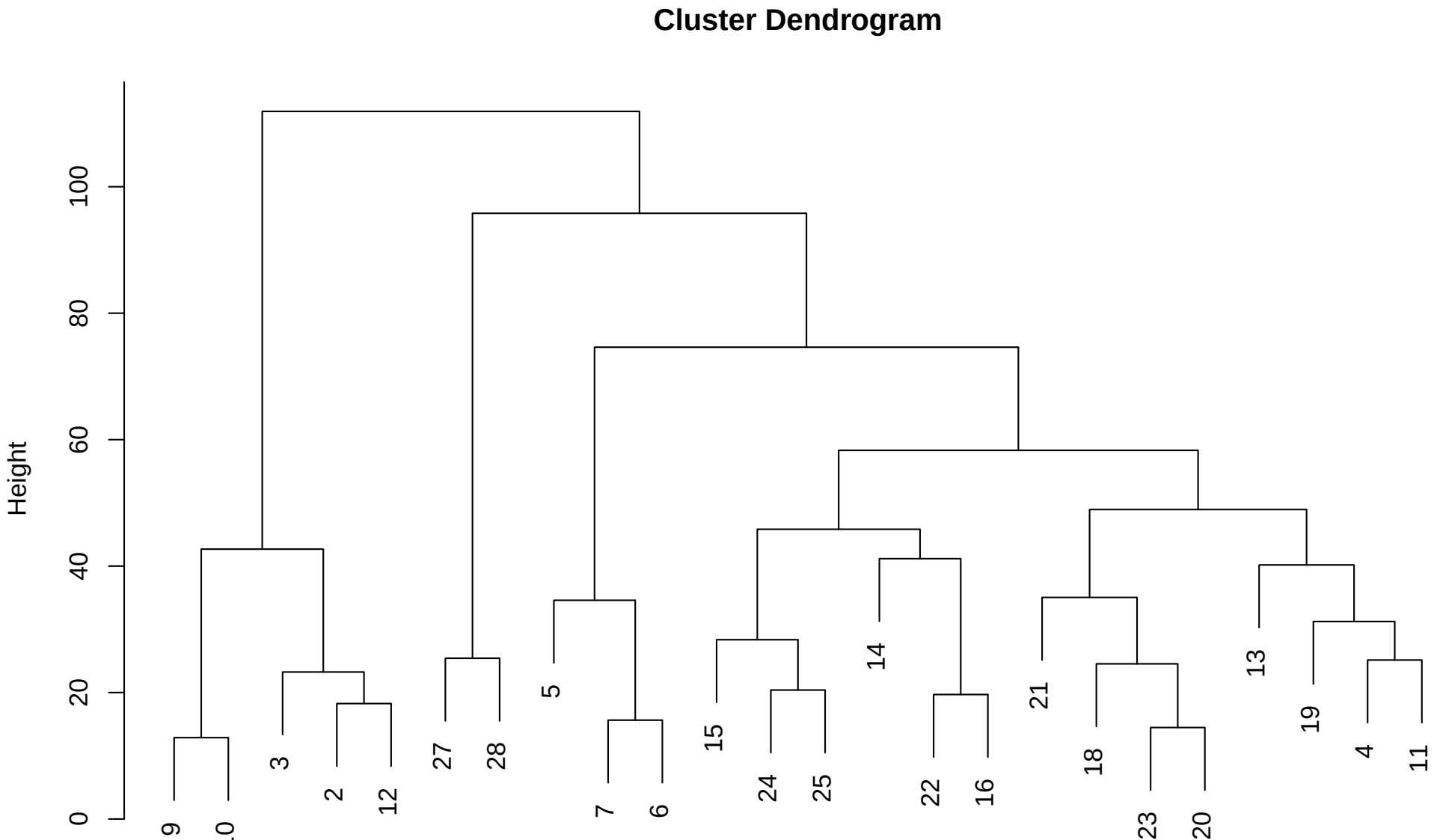
Apliquémosla sobre nuestra matriz previamente generada:

```
clusters<-hclust(eucl_dist, method = "complete")
```

Análisis de agrupamiento

Para visualizar nuestros clusters o mejor llamado dendograma usamos la función **plot()**:

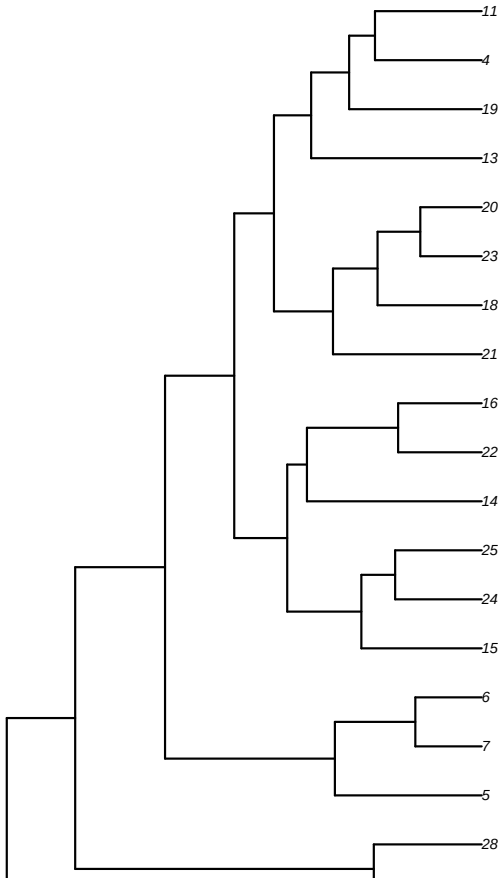
```
plot(clusters)
```

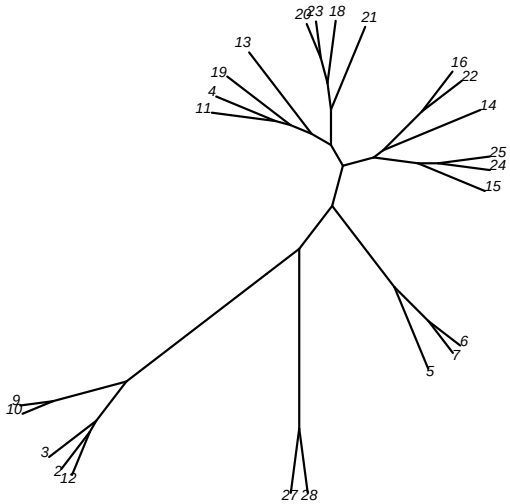


Análisis de agrupamiento

Con el paquete **ape()** podemos formatear de diferentes formas nuestro dendograma:

```
library("ape")
par( mfrow= c(1,3) )
plot(as.phylo(clusters), cex = 0.6)
plot(as.phylo(clusters), cex = 0.6, type = "unrooted")
plot(as.phylo(clusters), type = "fan")
```





Medida de la distorsión

Si nos interesa evaluar la bondad o qué tan bien nuestro dendograma explica nuestros datos entonces debemos medirlo a través de una matriz cofenética. En primer lugar se calcula la matriz cofenética con la función **cophenetic()** , que resulta de obtener una nueva matriz a partir del dendrograma y luego se calcula la correlación de Pearson entre la matriz cofenética y la matriz original.

```
mat.clusters<- cophenetic(clusters)
cor(mat.clusters, eucl_dist, method = "pearson")
```

```
## [1] 0.8052116
```

En este caso el coeficiente nos dio 0.8 lo cual está bien, entre más cercano a 1 nos dice qué tan bien correlacionados están y que mi dendograma explica en buena medida mis datos.

Análisis de ordenación canónica

Hay diversos tipos métodos de ordenación con los que podemos explorar nuestros datos ecológicos. R presenta diversas funciones para obtenerlas:

Método de Ordenación	Función	Paquete
PCA	princomp	stats
	prcomp	stats
	PCA	FactoMineR
	rda	vegan
PCoA	cmdscale	stats
	pcoa	ape
	wcmdscale	vegan
NMDS	**metaMDS**	vegan
	isoMDS	MASS
CA	CA	FactoMineR
	corresp	MASS
	cca	vegan

PCA

La función en vegan para PCA es rda(), que técnicamente significa Análisis de redundancia. No entraré en RDA (que al fin no tratamos previamente), pero cuando se ejecuta esta función sobre una matriz de especies sin ninguna variable ambiental, hace un PCA. En este ejemplo trabajaremos con otra data de ejemplo llamada **dune**, que son datos de vegetación de praderas de dunas, dunas, tienen valores de clase de cobertura de 30 especies en 20 sitios:

PCA

```
data("dune")
data("dune.env")
```

PCA

```
head(dune)
```

```
##   Achimill Agrostol Airaprae Alopgeni Anthodor Bellpere Bromhord Chenalb
## 1        1        0        0        0        0        0        0       0
## 2        3        0        0        2        0        3        4       0
## 3        0        4        0        7        0        2        0       0
## 4        0        8        0        2        0        2        3       0
## 5        2        0        0        0        4        2        2       0
## 6        2        0        0        0        3        0        0       0
##   Cirsarve Comapalu Eleopalu Elymrepe Empenigr Hyporadi Juncarti Juncbuf
## 1        0        0        0        4        0        0        0       0
## 2        0        0        0        4        0        0        0       0
## 3        0        0        0        4        0        0        0       0
## 4        2        0        0        4        0        0        0       0
## 5        0        0        0        4        0        0        0       0
## 6        0        0        0        0        0        0        0       0
##   Lolipere Planlanc Poaprat Poatriv Ranuflam Rumeacet Sagiproc Salirepe
## 1        7        0        4        2        0        0        0        0
## 2        5        0        4        7        0        0        0        0
## 3        6        0        5        6        0        0        0        0
## 4        5        0        4        5        0        0        5        0
## 5        2        5        2        6        0        5        0        0
```

PCA

```
head(dune.env)
```

```
##    A1 Moisture Management      Use Manure
## 1 2.8        1         SF Haypastu      4
## 2 3.5        1         BF Haypastu      2
## 3 4.3        2         SF Haypastu      4
## 4 4.2        2         SF Haypastu      4
## 5 6.3        1         HF Hayfield      2
## 6 4.3        1         HF Haypastu      2
```

PCA

Ahora corremos el PCA:

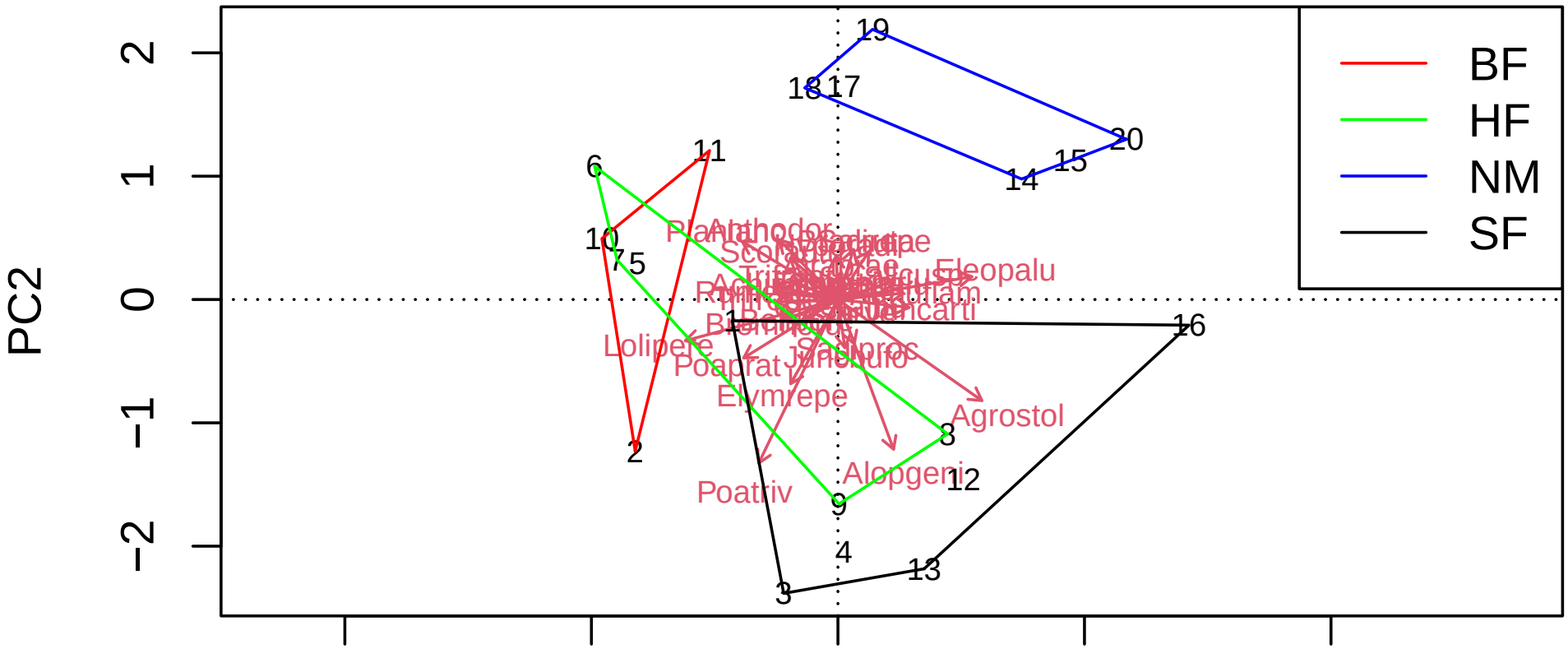
```
dune_pca <- rda(dune)
sum_dune_pca <- summary(dune_pca)
head(sum_dune_pca)
```

```
##
## Call:
## rda(X = dune)
##
## Partitioning of variance:
##               Inertia Proportion
## Total           84.12          1
## Unconstrained   84.12          1
##
## Eigenvalues, and their contribution to the variance
##
## Importance of components:
##                             PC1     PC2     PC3     PC4    PC5     PC6
## Eigenvalue              24.7953 18.1466 7.62913 7.15277 5.6950 4.33331 3.1
## Proportion Explained     0.2947  0.2157 0.09069 0.08503 0.0677 0.05151 0.0
## Cumulative Proportion    0.2947  0.5105 0.60115 0.68618 0.7539 0.80539 0.8
```

PCA

Para visualizarlo usamos la función **biplot()**

```
biplot(dune_pca)
ordihull(dune_pca, groups = dune.env$Management,
         col = c("red", "green", "blue", "black"))
man_names<- levels(dune.env$Management)
legend("topright",col = c("red", "green", "blue", "black"),
       lty = 1, legend = man_names )
```



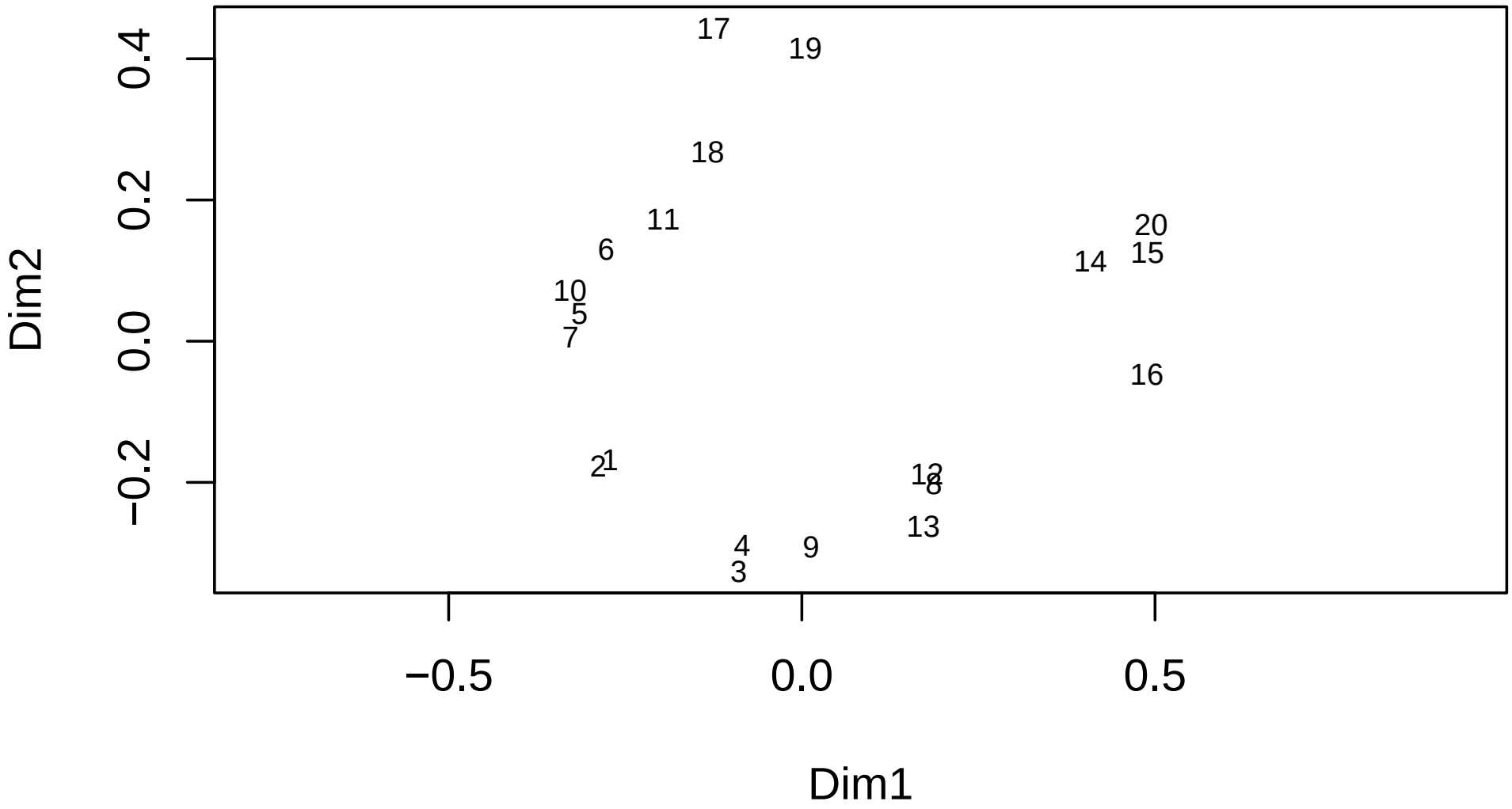
PCoA

Usaremos la misma matriz para calcular PCoA y dibujar el diagrama de ordenación. Recuerden que la ventaje de este método es poder uscar cualquier tipo de matriz de distancia que deseemos:

```
d <- vegdist(dune, method = "jaccard")
ord <- wcmdscale(d, eig = TRUE)
```

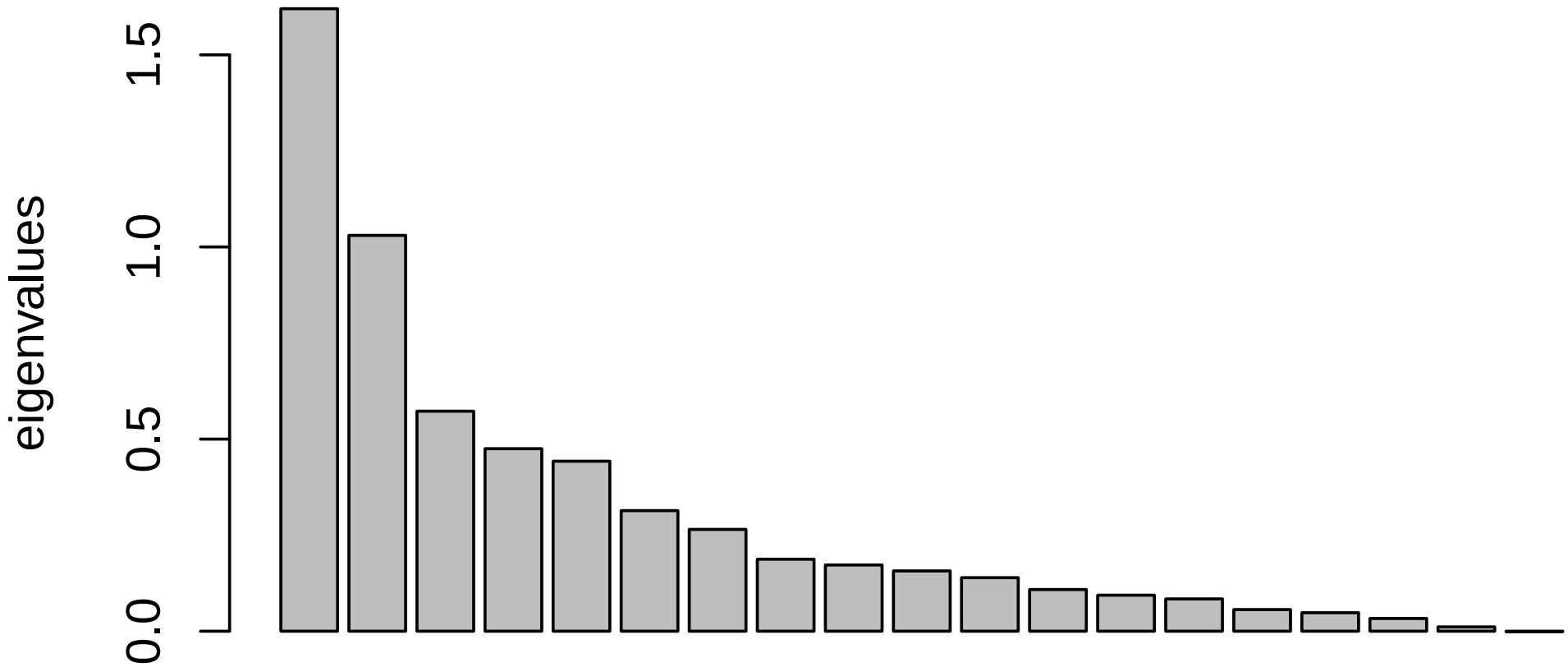
PCoA

```
ordiplot(ord, display = 'sites', type = 'text')
```



PCoA

```
barplot (ord$eig, las = 3, ylab = 'eigenvalues')
```



NMDS

El objetivo de NMDS es colapsar la información de múltiples dimensiones (p. ej., de múltiples comunidades, sitios, etc.) en solo unas pocas, para que puedan visualizarse e interpretarse.

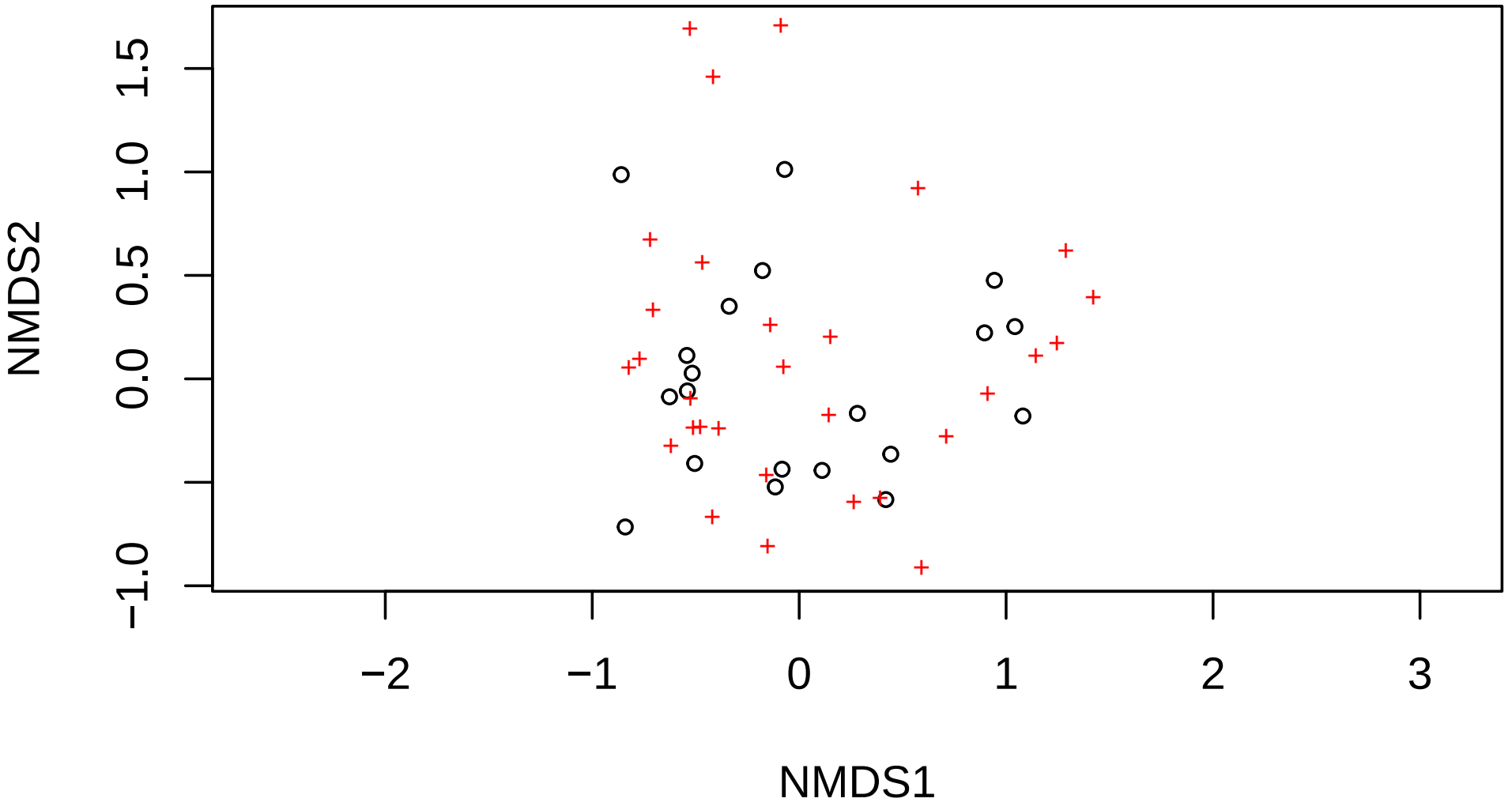
```
nmds <- metaMDS(dune, distance = "bray", k = 2)
```

```
## Run 0 stress 0.1192678
## Run 1 stress 0.1183186
## ... New best solution
## ... Procrustes: rmse 0.02026955  max resid 0.06495869
## Run 2 stress 0.1192679
## Run 3 stress 0.1183186
## ... Procrustes: rmse 9.525465e-06  max resid 3.036745e-05
## ... Similar to previous best
## Run 4 stress 0.1939202
## Run 5 stress 0.1183186
## ... New best solution
## ... Procrustes: rmse 7.381406e-06  max resid 2.41098e-05
## ... Similar to previous best
## Run 6 stress 0.1183186
## ... Procrustes: rmse 1.5004e-05  max resid 4.817011e-05
```

NMDS

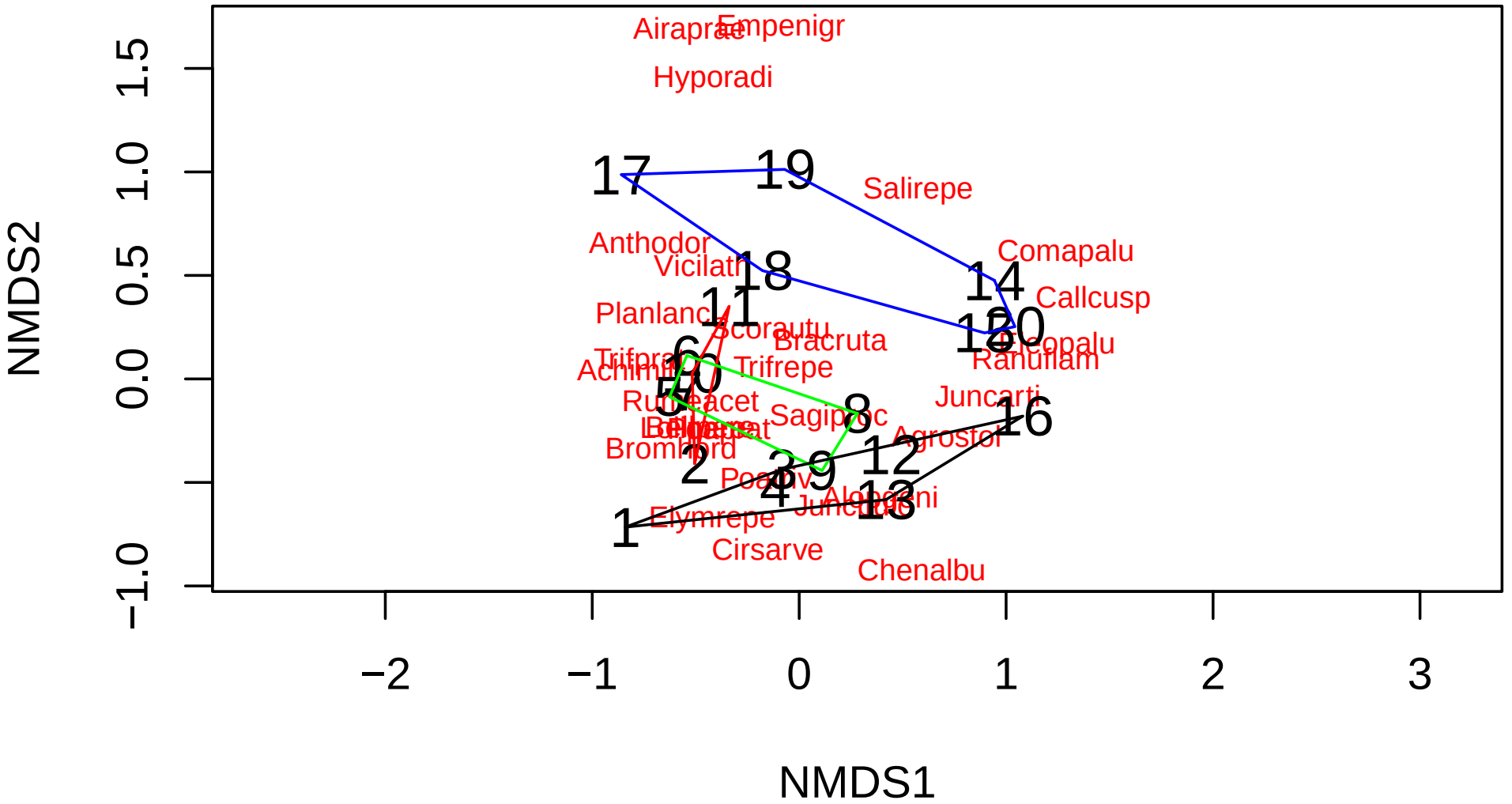
Visualizando:

```
plot(nmds)
```



NMDS

```
ordiplot(nmds,type="n")
orditorp(nmds,display="species",col="red",air=0.01)
orditorp(nmds,display="sites",cex=1.25,air=0.01)
ordihull(nmds, groups = dune.env$Management, col = c("red", "green", "blue"
```



NMDS

De esta manera visualizamos nuestros datos o coordenadas:

```
head(scores(nmds, display = "species"))
```

```
##                NMDS1       NMDS2
## Achimill -0.8228068  0.04327339
## Agrostol  0.7109646 -0.28923693
## Airaprae -0.5282192  1.67986433
## Alopgeni  0.3909736 -0.58595489
## Anthodor -0.7202345  0.65913519
## Bellpere -0.4783773 -0.24447320
```

```
head(scores(nmds, display = "sites"))
```

```
##        NMDS1       NMDS2
## 1 -0.8405272 -0.71585098
## 2 -0.5048507 -0.40894087
## 3 -0.0826648 -0.43667799
## 4 -0.1156159 -0.52224373
## 5 -0.6265428 -0.08670114
## 6 -0.5426932  0.11315163
```

CCA

También conocido como análisis de correspondencia canónica.

Los métodos de ordenación anteriores son ordenaciones “sin restricciones”, lo que significa que la ordenación se realiza solo considerando los recuentos de especies (counts).

La ordenación restringida es apropiada para lo que es común en los datos ecológicos: una matriz de comunidades y otra matriz de características ambientales o fisicoquímicas.

Con la ordenación restringida, podemos preguntarnos cómo se relacionan las variables ambientales con la composición de la comunidad.

CCA

Veámoslo con un ejemplo:

```
data("varespec") #especies
data("varechem") #fisicoquímicos
```

```
varespec[1:4, 1:4]
```

```
##    Callvulg Empenigr Rhodtome Vaccmyrt
## 18     0.55    11.13     0.00     0.00
## 15     0.67     0.17     0.00     0.35
## 24     0.10     1.55     0.00     0.00
## 27     0.00    15.13     2.42     5.92
```

```
varechem[1:4, 1:4]
```

```
##       N    P     K    Ca
## 18 19.8 42.1 139.9 519.4
## 15 13.4 39.1 167.3 356.7
## 24 20.2 67.7 207.1 973.3
## 27 20.6 60.8 233.7 834.0
```

CCA

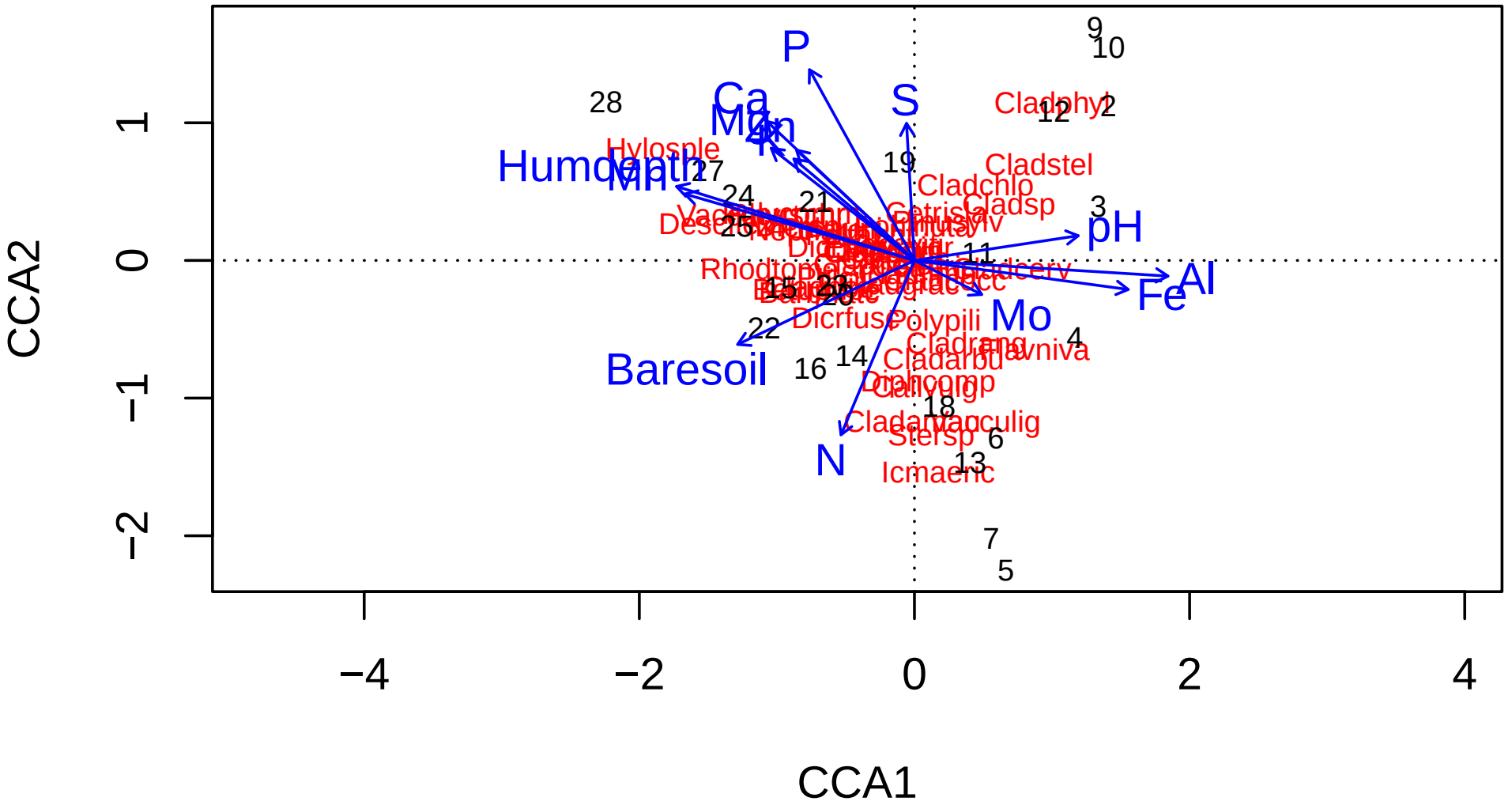
```
vares_cca <- cca(varespec ~ N+P+K+Ca+Mg+S+Al+Fe+Mn+Zn+Mo+Baresoil+Humdepth+
                 data=varechem)
summary(vares_cca)
```

```
##
## Call:
## cca(formula = varespec ~ N + P + K + Ca + Mg + S + Al + Fe +      Mn + Z
##
## Partitioning of scaled Chi-square:
##               Inertia Proportion
## Total          2.0832      1.000
## Constrained    1.4415      0.692
## Unconstrained  0.6417      0.308
##
## Eigenvalues, and their contribution to the scaled Chi-square
##
## Importance of components:
##                          CCA1   CCA2    CCA3    CCA4    CCA5    CCA6    C
## Eigenvalue             0.4389 0.2918 0.16285 0.14213 0.11795 0.08903 0.07
## Proportion Explained   0.2107 0.1401 0.07817 0.06823 0.05662 0.04274 0.03
## Cumulative Proportion 0.2107 0.3507 0.42890 0.49713 0.55375 0.59649 0.63
##                          CCA8    CCA9    CCA10    CCA11    CCA12    CCA1
```

CCA

Ahora, visualicemos los resultados en un biplot:

```
plot(vares_cca)
```



CCA

Si queremos saver cuales son las variables con más peso en el análisis ocupamos la función **envfit()**.

```
envfit(vares_cca  ~ N+P+K+Ca+Mg+S+Al+Fe+Mn+Zn+Mo+Baresoil+Humdepth+pH ,
       data=varechem )
```

```
##
## ***VECTORS
##
##              CCA1     CCA2     r2 Pr(>r)
## N         -0.36715 -0.93016 0.2781  0.025 *
## P         -0.49393  0.86950 0.3861  0.008 **
## K         -0.76057  0.64926 0.2028  0.089 .
## Ca        -0.72500  0.68875 0.3355  0.012 *
## Mg        -0.78169  0.62367 0.2703  0.036 *
## S         -0.08664  0.99624 0.1505  0.184
## Al         0.99584 -0.09111 0.5130  0.002 **
## Fe         0.98659 -0.16322 0.3716  0.014 *
## Mn        -0.94705  0.32108 0.5003  0.001 ***
## Zn        -0.72738  0.68624 0.2133  0.086 .
## Mo         0.88340 -0.46863 0.0462  0.607
## Baresoil -0.91064 -0.41319 0.2960  0.035 *
## Humdepth -0.95304  0.30286 0.4594  0.004 **
```

perMANOVA

Análisis de varianza multivariante permutacional (perMANOVA).

La evaluación de las diferencias en la composición de la comunidad se realiza con el análisis de varianza multivariado permutacional.

Estas pruebas se realizan sobre distancias, es decir, evalúan las diferencias entre comunidades en función de la disimilitud.

perMANOVA

```
dune_perm <- adonis(dune ~ Management+Use+Moisture,
                    data = dune.env, method = "euclidean")
dune_perm2 <- adonis(dist(dune) ~ Management+Use+Moisture,
                     data = dune.env)
```

perMANOVA

```
dune_perm;dune_perm2
```

```
##
## Call:
## adonis(formula = dune ~ Management + Use + Moisture, data = dune.env,
##
## Permutation: free
## Number of permutations: 999
##
## Terms added sequentially (first to last)
##
##             Df SumsOfSqs MeanSqs F.Model      R2 Pr(>F)
## Management  3    555.38 185.128  3.2580 0.34747  0.001 ***
## Use         2    136.15  68.077  1.1981 0.08518  0.270
## Moisture    3    281.76  93.919  1.6528 0.17628  0.049 *
## Residuals  11    625.05  56.823         0.39106
## Total      19   1598.35                1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

##
## Call:
## adonis(formula = dist(dune) ~ Management + Use + Moisture, data = dune.e
```