Geobotánica, Práctica 2

Clasificación y ordenación con R

Dr. Francisco José Alcaraz Ariza
Universidad de Murcia
España

(versión de 14 de febrero de 2013)

Índice

Índice de cuadros

Índice de figuras

Geobotánica, Práctica 2

Clasificación y ordenación con R

1. Introducción

1.1. Qué es R

R es un lenguaje y entorno para el tratamiento numérico de datos y la creación de gráficos. R implementa un dialecto del lenguaje S y provee una amplia variedad de técnicas estadísticas y gráficas (modelización linear y no linear, tests estadísticos, análisis de series temporales, clasificación, ordenación, etc.).

R es fácilmente extensible al permitir definir nuevas funciones e incluso bibliotecas (librerías), de las que existen numerosas ya disponibles.

R es software libre bajo la licencia GPL, por lo que puede obtenerse el código completo de la aplicación. Está disponible para numerosas plataformas y sistemas operativos, incluidos Solaris, Windows 9x/NT/2000/XP, FreeBSD y Linux.

R proporciona:

- Un conjunto coherente y extensivo de instrumentos para el análisis y el tratamiento estadístico de datos.
- Un lenguaje para expresar modelos estadísticos y herramientas para manejar modelos lineales y no lineales.
- Utilidades gráficas para el análisis de datos y la visualización en cualquier estación gráfica o impresora.
- Un eficiente lenguaje de programación orientado a objetos, que crece fácilmente merced a la comunidad de usuarios.

En la distribución base de R se incluyen herramientas para:

- La descripción, tabulación y representación gráfica de datos.
- Inferencia estadística. Regresión y análisis de la varianza. Modelos lineales generalizados.
- Técnicas multivariantes: clasificación y ordenación.
- Series temporales. Análisis de supervivencia.
- Cálculo matricial.
- Resolución de sistemas de ecuaciones lineales.
- Cálculo numérico.
- Interpolación.

Pero además, existen librerías que incorporan análisis y funciones adicionales, destacando algunas relativas a:

- Interpolación espacial.
- Análisis discriminante.
- Visualización de datos en tres dimensiones.
- Análisis de la vegetación (vegan)

1.2. Descarga e instalación

Descarga en **`http://cran.r-project.org/bin`**, muchas distribuciones Linux tienen en sus repositorios el paquete básico de R.

El sistema se puede extender con librerías (paquetes de funciones), que incluyen funciones con documentación en un formato estandarizado, accesible con la ayuda normal de «R» una vez instaladas.

Muchas librerías de interés:

- Vegan (análisis de vegetación y taxonómico)
- RSvgDevide (salidas gráficas en formato svg)

- scatterplot3d (salidas en tres dimensiones, muchas animadas)
- clim.pac, climatol (análisis de datos climáticos, diagramas, etc.).
- rgl (acceso a librerías OpenGL)

Estas librerías se instalan según el sistema operativo:

- Windows: desde el menú de paquetes (librerías)
- Linux, desde fuera del programa: **`R CMD INSTALL paquete-x.y.z.tar.gz`**
- Linux, desde dentro de R: **`install.packages (nombre-del-paquete)`**

1.3. Empezando la sesión

En estas prácticas se trabajará con un servidor que centralizará datos y programas para todos los usuarios. En realidad trabajaremos todos a la vez en un mismo ordenador que no está en la microaula; nos conectaremos al mismo, de modo que las máquinas de la microaula funcionarán como terminales.

El servidor se llama **servbiob.inf.um.es** y funciona bajo sistema operativo Linux. En un sistema operativo multiusuario como este resulta frecuente hacer conexiones remotas desde otro ordenador que actúa como terminal. Existen diversos programas para hacer esta conexión, vamos a utilizar VNC (Virtual Network Client), para conectarse sigua los siguientes pasos:

1. Arranque desde el menú de Windows → Prácticas → VNC-> Vncviewer
2. Introduzca el nombre del servidor (**`servbiob.inf.um.es`**) y haga click en aceptar (OK)
3. Si todo ha ido bien verán la pantalla de bienvenida de servbiob.bio.um.es donde deberán introducir
    4. Su nombre de usuario (**`ma`**<x[1]>)
    5. Su contraseña (**`biologia`**<x>)
6. Si lo anterior se ha realizado correctamente el sistema mostrará la pantalla del escritorio en el servidor remoto (algo similar a lo mostrado en la figura 1).
7. Pulse entonces en el icono terminal que aparece en la barra de herramientas superior.
8. En la terminal escribiremos **`cd Desktop/Botanica/R`** para pasar al directorio donde están ubicadas las funciones que vamos a utilizar durante la práctica. Ahora puede arrancar el programa simplemente tecleando **`R`**.

1.4. Entrada y abandono de R

Para arrancar R, basta con invocarlo desde la línea de comandos (Linux) o haciendo doble «click» sobre el icono *R* en el escritorio de Windows. En el caso de los usuarios de Windows se abrirá una ventana desde la que se podrán ir escribiendo comandos, seleccionando datos, etc. El usuario de Linux podrá escribir comandos desde la misma consola en la que inició R.

Para terminar una sesión de R adecuadamente debe utilizar la función «q()» que obliga a determinar si queremos abandonar los datos y el histórico de órdenes, lo deseamos grabar o cancelamos el abandono.

Es fundamental al iniciar la sesión indicar cuál será el directorio de trabajo, en el que se encontrarán todos los ficheros a cargar y donde se grabarán los que se creen durante la sesión. Desde Windows en el menú «Archivo»->Cambiar directorio; en Linux con la orden «setwd("directorio-de-trabajo")».

1.5. Una sesión de trabajo sencilla

- **`x <- rnorm(25)`**: creamos un vector de 25 elementos con valores aleatorios.
- **`x`**: muestra los valores del vector creado.
- **`a <- rnorm(25)`**: creamos otro vector de 25 elementos.

1 <x> es un número, del 1 al 20, que será señalado para cada puesto informático por el profesor.

- **`y <- x + a/10`**: suma ponderada de los vectores anteriores elemento a elemento.
- **`rm(a)`**: elimina el vector a.
- **`summary(x)`**: estadísticos de x.
- **`summary(y)`**: estadísticos de y.
- **`mlyx <- lm(y~x)`**: análisis de regresión para «x», independiente «y» dependiente (en Windows el símbolo ~ se obtiene manteniendo pulsada la tecla «ALT» y tecleando «126», aquí (Linux) con «ALTGR» + «4».
- **`plot (x,y)`**: representación de los puntos analizados.
- **`abline(mlyx)`**: representación de la recta obtenida.
- **`summary(mlyx)`**: representación de los resultados del análisis.
- **`plot(mlyx)`**: representación de los distintos elementos resultantes del análisis.
-

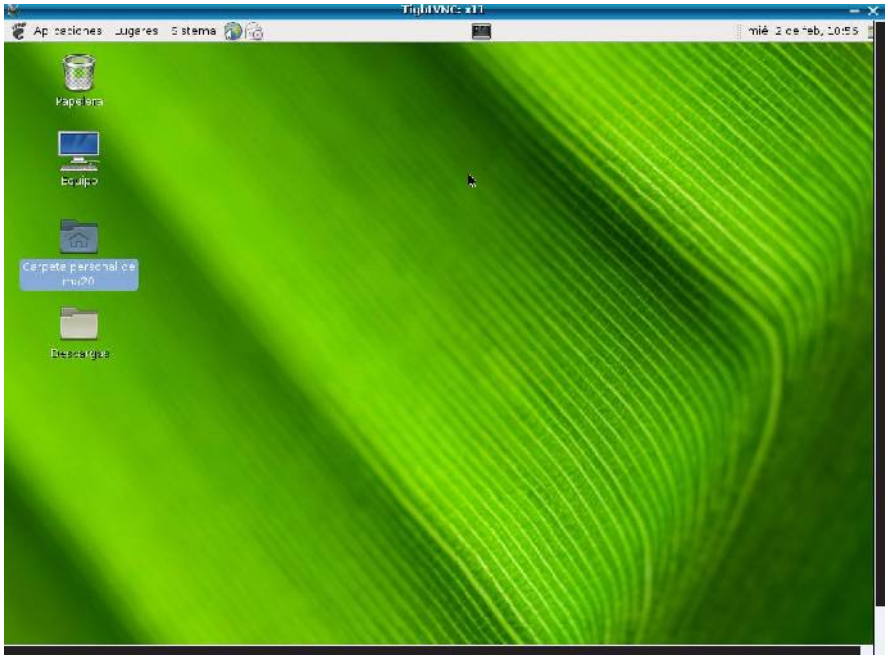

Figura 1: Pantalla en servbiob.inf.um.es

1.6. Ayuda en R

- **`help(), ?`** : Proporciona ayuda sobre una palabra clave.
- **`help.start()`**: Inicia consulta de la ayuda desde un navegador.
- **`example()`**: Muestra los resultados propuestos en el ejemplo.
- **`demo()`**: Muestra la relación de *demos* disponibles.
- **`library()`**: Muestra la relación de bibliotecas de funciones disponibles.
- **`data()`**: Muestra la relación de datos de ejemplo disponibles.
- **`apropos()`**: Muestra la relación de objetos disponibles con una cadena dada.

2. Preparación de los datos

2.1. Introducción

En vegetación se suele trabajar con tablas, en las que las variables (especies) se sitúan en filas, pero en la ortodoxia estadística las variables debieran ubicarse en columnas; de hecho así es como esperan encontrar los datos muchas de las funciones que se van a utilizar (las unidades muestrales deben ubicarse en las filas); no obstante esto no supone un gran problema, pues una vez cargados los datos en «R» la función **`t(matriz)`** transpone la matriz y ubica las especies en la posición adecuada.

2.2. Los nombres de las especies

También conviene acortar los nombres de las especies, para evitar gráficos colapsados por las etiquetas. La abreviación estandarizada es la CEP (Cornell Ecology Programas), en la que se usan ocho caracteres: en primer lugar los cuatro primeros del nombre genérico y después los 4 primeros del epíteto específico o, en su caso, subespecífico o varietal. Por ejemplo, *Quercus ilex* subsp. *ballota* se escribiría como **QUERILEB**. En la librería vegan se dispone de la función **`make.cepnames`** para transformar largas listas de nombres de organismos al estándar.

2.3. La dominancia-abundancia de Braun-Blanquet

Los valores para cada especie en cada unidad muestral deben ser numéricos. Esto plantea algunos problemas con dos índices habituales en los inventarios (unidades muestrales) fitosociológicos, **+, r**; además se sabe que estadísticamente la mayor parte de las especies tienen como índice de abundancia el «2». Por lo tanto es necesaria una transformación para resolver lo primero y una estandarización para que todas las clases de abundancia-dominancia tengan similar importancia en el análisis. La transformación-estandarización más aceptada es la de Van deer Maarel (cuadro 1).

Cuadro 1: Transformación-estandarización de van deer Maarel del índice de abundancia-dominancia

Índice	Valor
ausencia	0
r	1
+	2
1	3
2a	4
2b	5
2c	6
3	7
4	8
5	9

2.4. Datos de la comunidad y ambientales

- Mejor separarlos en dos ficheros distintos
- El orden de los lugares debe ser idéntico en ambos conjuntos de datos
- Las variables tipo factor deben ser codificadas con nombres informativos
- No usar variables tipo «dummy»
- Los datos de las especies deben ser numéricos: use «0» para las especies ausentes, mientras que valores ambientales ausentes deben de ponerse como «NA»

2.5. Transformación de los datos

En ocasiones es necesario realizar transformaciones de los datos para sacar más luz a los mismos; estas transformaciones pueden ser previas a los análisis o posteriores, como consecuencia de los mismos. Si un análisis inicial revela que varias especies o unidades muestrales están sobrevaloradas, interesa entonces homogeneizar la importancia de todas las variables, para ello se puede:

- **Estandarizar:** `scale(matriz)`, si se producen números negativos habrá que sumarle a la matriz un número positivo superior al negativo más bajo (`matriz + 0.5->matriz`).
- **Enmascarar** (eliminar filas o columnas de la matriz):
  - `x1<- x[-1,]`: elimina la unidad muestral (fila) 1 de «x»
  - `x1<- x[,-2]`: elimina la especie (columna) 2 de «x»
- **Transformar a presencia/ausencia** (ejemplo para la matriz yesos):
  - `yesos2 <- ifelse(yesos>0,1,0)`

2.6. Creación de un fichero simple

Vamos a crear un ejemplo sencillo de fichero para «R» con un editor de textos sencillo (bloc de notas, kwriter, nedit, mc, emax, etc.). Se dispone de cinco unidades muestrales de vegetación correspondientes a encinares y alcornocales, abreviaremos los nombres (ver cuadro 2) y grabaremos el archivo con el nombre de «**`encinar.txt`**» (ver cuadro 3).

Para archivos más voluminosos es aconsejable usar una hoja de cálculo (OpenOffice.org Calc, Excel, Lotus, etc.). Al final habrá que exportar la hoja a un formato de «solo texto», siendo uno de los más útiles el formato «.csv». Una vez exportado conviene abrir el fichero resultante con un simple editor de textos para comprobar que en el proceso no se hallan incluido símbolos extraños ni comas o punto y comas fuera de sitio. En la exportación se puede elegir el tipo de separador (espacio, coma, punto y coma) entre datos, depende de cuál sea nuestra elección la función para cargar los archivos por parte de «R» (read.table o read.csv) puede necesitar unos parámetros u otros.

Cuadro 2: Ejemplos sobre la forma de abreviar los nombres de los taxones

Nombre científico	Abreviatura
Astragalus lusitanicus	ASTRLUSI
Cistus clusi subsp. *multiflorus*	CISTMULT
Quercus ilex subsp. *ballota*	QUERILEB
Quercus suber	QUERSUBE

Cuadro 3: Ejemplo de 5 muestras hipotéticas de encinar y alcornocal

QUERILEB	QUERSUBE	CISTMULT	CISTPOPU	ERICARBO	LAVASAMP	LAVALUIS
3	0	3	0	0	2	0
5	1	3	1	1	3	1
1	5	0	2	2	0	3
0	4	0	3	2	0	3
5	0	4	6	0	2	0

Vamos a crear el cuadro ejemplo fichero de las dos maneras, con un editor de texto sencillo (ver figura 2) y con una hoja de cálculo. En el primer caso se deberán teclear los datos en el formato mencionado (separando con un espacio uno de otro y cada muestra en líneas diferentes), tal y como aparece en la figura mc. El fichero será guardado con el nombre de «**`encinar.txt`**».

En el segundo caso cada dato ocupará una celdilla, la celdilla A1 debe quedar en blanco si queremos usar la primera columna para poner etiquetas a las distintas unidades muestrales.

Se cargará el fichero (**read.table** o r**ead.csv**) y después con **edit(variable en la que se ha grabado el contenido del fichero)** podremos comprobar si la carga ha sido correcta.

En el caso de haber realizado el fichero con la hoja de cálculo, habrá que hacer la exportación a «**.csv**», luego habrá que editar el fichero para borrar el «;» o la «,» inicial antes de que lo pueda leer «R». También hay que cambiar la forma en que «R» lee el fichero, pues la separación entre dato y dato en la misma línea no es ahora un espacio, sino una «,» (OpenOffice.org Calc) o un «;» (Excel); de modo que el comando para cargar con Calc los datos será:

```
read.table("encinar.txt",header=T) -> encinar
```

o bien:

```
read.table("encinar.txt",header=T,sep=";") -> encinar
```

o bien:

```
read.csv("yesos.dat")->yesos
```

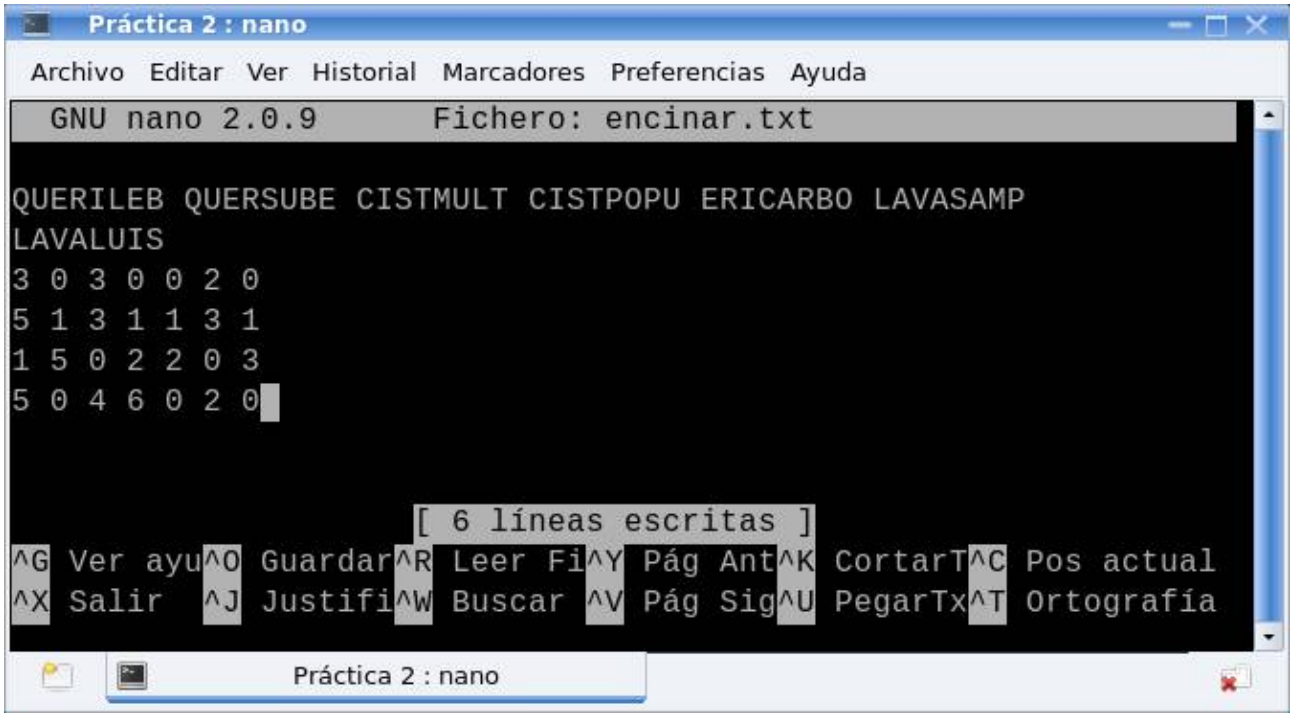

Figura 2: El fichero encinar.txt realizado con el editor de texto nano

3. Clasificación con R

3.1. Introducción

R incluye la función **hclust** para la realización de clasificaciones jerárquicas. La librería **vegan** no está enfocada hacia la clasificación, pero proporciona alguna herramienta útil para este tipo de análisis numérico (**vegdist**). La librería **labdsv** es particularmente potente en sus funciones de clasificación.

Una clasificación precisa que el investigador le pase dos opciones básicas a R:

- Medida de similitud/disimilitud para calcular la matriz de distancias
- Estrategia de fusión

Posteriormente, en la interpretación de los datos el investigador deberá decidir el punto de corte y, con ello, el número de grupos.

Se puede usar la función nativa de R **dist** para calcular la matriz de distancias; por defecto el programa usa la distancia euclídea (euclidean), pero es posible elegir varias más con la opción **method="<método>": maximum, manhattan, canberra, binary, minkowski**. La librería «**vegan**», que usaremos más adelante en la

práctica, tiene una función «**vegdist**» que incluye varios tipos de distancia (**euclidean, manhattan, gower, canberra, *bray*, kulczynski, morisita, horn, binomial, jaccard, mountford, raup, binomial, chao**).

Las estrategias de fusión disponibles en **hclust** son las siguientes:

- **"ward"**
- **"single"**: o vecino más próximo, encuentra el árbol más corto que incluya a todos los puntos, encuentra discontinuidades y encadena grupos de tamaño desigual.
- **"complete"**: crea grupos compactos de tamaño similar, incluso aunque no se den en la realidad.
- **"average"(UPGMA**: entre los dos métodos anteriores, minimiza la correlación cofenética).
- **"mcquitty"**
- **"median"**
- **"centroid"**

3.2. Clasificación jerárquica

La función «hclust» permite realizar clasificaciones jerárquicas. Vamos a realizar una prueba con el fichero «encinar.txt» creado en el apartado anterior. (ver cuadro 4); el resultado se puede observar en la figura 3.

Cuadro 4: Pasos para una clasificación jerárquica con matriz de distancia euclídea y estrategia de fusión de la media

Etapas	Comando
R lee el fichero de datos y lo asigna a la matriz «encinar»	`read.table("encinar.txt",header=T) -> encinar`
Clasificación jerárquica con estrategia de fusión de la media el resultado se asigna a «hc»	`hc <- hclust(dist(encinar), method="average")`
Finalmente, se muestra el gráfico	`plot(hc)`
Vea como cambia ahora con esta orden	`plot(hc,hang=-1)`

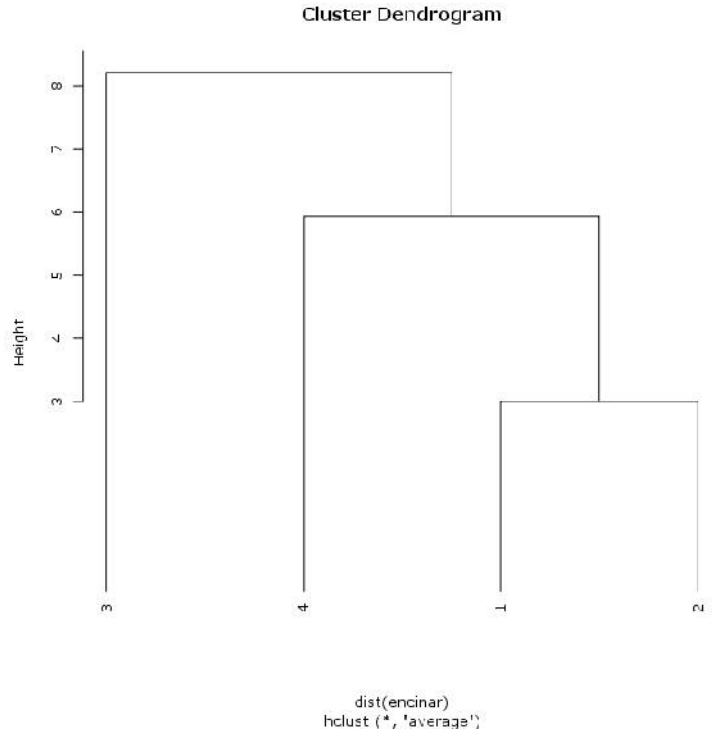

Figura 3: Clasificación de las muestras incluidas en encinar.txt con la estrategia de fusión de la media

La orden **plot** es muy potente en R, por ejemplo el tamaño de los textos se puede controlar del modo siguiente:

- **cex**: tamaño del texto (es un multiplicador, para textos menores, usar números menor de 1)
- **cex.axis**: tamaño de las etiquetas tick del eje.
- **cex.lab**: tamaño de las etiquetas del eje
- **cex.main**: tamaño del texto título del gráfico
- **cex.sub**: tamaño del texto del subtítulo.

Si el texto ha quedado muy pequeño la orden para dibujar el cluster puede ser modificada así:

```
plot(hc,hang=-1,cex=1.5)
```

Tras este ejemplo, se va a utilizar un fichero previamente creado con unas 61 especies y 29 unidades muestrales de vegetación correspondientes a tomillares abiertos en gipsisoles compactos del Sureste de España (**yesos.dat**).

Se aplicarán diversas transformaciones y funciones a la matriz de datos, para de este modo dar una idea general de las posibilidades de R aplicado al análisis numérico de datos sobre vegetación. La mayoría de las funciones que van a utilizar tienen numerosas opciones, de las que se va a usar sólo las más generales. Se puede profundizar en el tema descargando el manual de referencia de R (ver fuentes bibliográficas, documentos) en cuyas páginas 895 a 925 se detalla la biblioteca básica «stats» que incluye muchas de estas funciones de clasificación.

Tras entrar en «R» fijamos como directorio de trabajo el que contiene el fichero de tomillares de yesos, a continuación se leen los datos de **yesos.dat**, asignádolos a la matriz «**yesos**»:

```
read.table("yesos.dat",header=T)->yesos
```

Si quieren ver el contenido de la matriz «yesos» basta con teclear desde «R»:

```
yesos
```

o

```
edit(yesos)
```

Para saber sus dimensiones: **dim(yesos)**. Los nombres abreviados de las especies los obtenemos con **names(yesos)**. Las muestras simplemente están numeradas, pero de haber tenido etiquetas se podrían ver con **row.names(yesos)**.

Ahora vamos a realizar varias clasificaciones jerárquicas usando el fichero yesos:

```
hc <- hclust(dist(yesos,"manhattan"),"average")
plot(hc,hang=-1)
plot(hclust(vegdist(yesos,"jaccard"),method="average"),hang=-1)
```

Realice pruebas cambiando la estrategia de fusión (**war, sin, com, mcq, med,cen**)[2] .

Vamos a ver la forma de tener muchas clasificaciones diferentes en un solo gráfico:

```
layout(matrix(1:16,4,4))

distancia<-c("euclidean", "manhattan", "gower", "canberra", "bray",
"kulczynski", "morisita", "horn", "binomial", "jaccard", "mountford",
"raup", "chao")

for(i in 1:13)plot(hclust(vegdist(yesos,distancia[i]),"war"),hang=-1)
```

2 Utilizando las flechas del cursor para arriba o para abajo se repiten las órdenes utilizadas previamente, se puede hacer aparecer el primer comando y cambiar «ave» por cualquiera de las otras abreviaturas

Podemos hacer más pruebas usando uno de los ficheros de datos incluidos en la librería «vegan»:

```
library(vegan)
data(dune)
dis<-vegdist(dune)
clust<-hclust(dis,"single")
plot(clus)
```

Algunos autores prefieren como estrategia de fusión la del vecino más próximo (**single**), por que puede ser representada muy bien en las ordenaciones y permite encontrar discontinuidades en los datos. Sin embargo es proclive a encadenar datos. La estrategia de fusión del vecino más alejado (**complete**) origina cluster compactos. Muchos investigadores prefieren la estrategia de fusión de la media (**average**), compromiso entre los dos extremos anteriores y más neutral en la agrupación. El método **average** es a veces conocido como ***UPGMA***, muy popular en los inicios de la genética. Todos estos métodos son aglomerativos.

3.3. Visualización e interpretación de las clases

Una vez obtenido el cluster, hay que decidir cuáles son los grupos que se reconocen y eso depende de los puntos de corte que se elijan.

Entre las librerías base de «R» se incluyen las funciones **rect.hclust** para ver los cortes y la función **cutree** para hacer una clasificación con un cierto número de clases (ver figura 4):

```
library(vegan)
data(dune)
dis<-vegdist(dune)
cluc<-hclust(dis,"complete")
plot(cluc)
rect.hclust(cluc,3)
grp<-cutree(cluc,3)
```

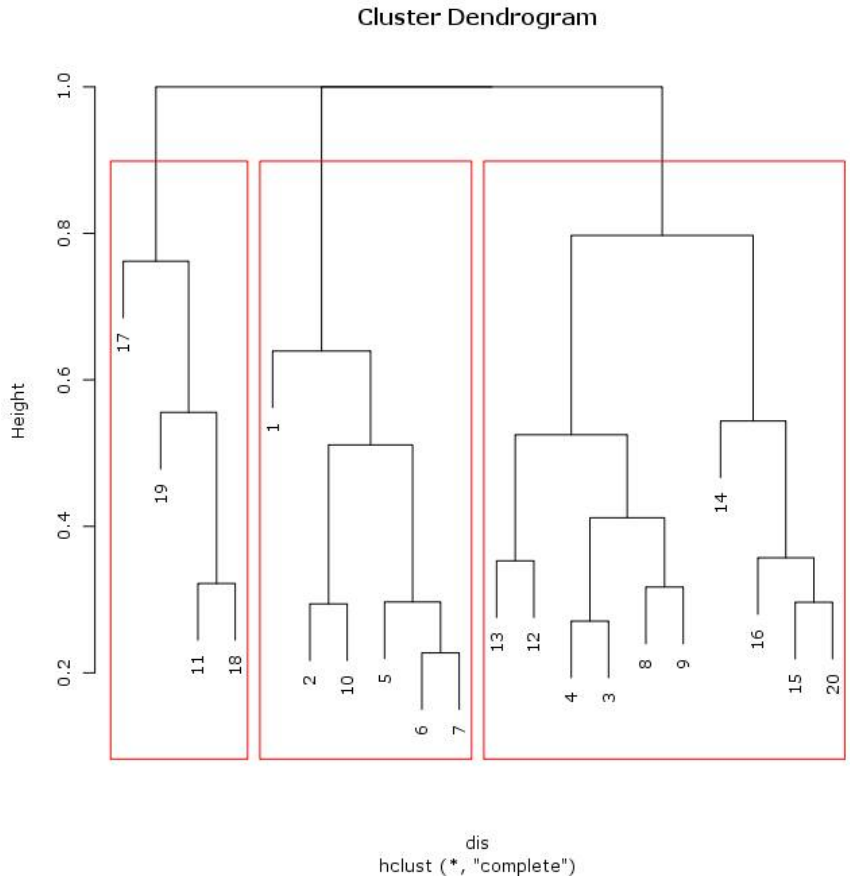

Figura 4: Ejemplo de la señalización de grupos usando rect.hclust

Una forma natural de inspeccionar la bondad de una clasificación de comunidades es comprobar hasta qué punto permite predecir variables ambientales externas que no fueron usadas en la clasificación.

Los resultados de la clasificación (cluster) pueden ser mostrados en diagramas de ordenación; vegan dispone de varias funciones apropiadas para esto: **ordihull, ordispider, ordiellipse**. Por ejemplo, para aplicar los tres grupos que hemos propuesto en las líneas anteriores y grabado en «grp» (ver figura 5):

```
library(vegan)
data(dune)
ord<-cca(dune)
plot(ord,display="sites")
ordihull(ord,grp, lty=2, col="red")
```

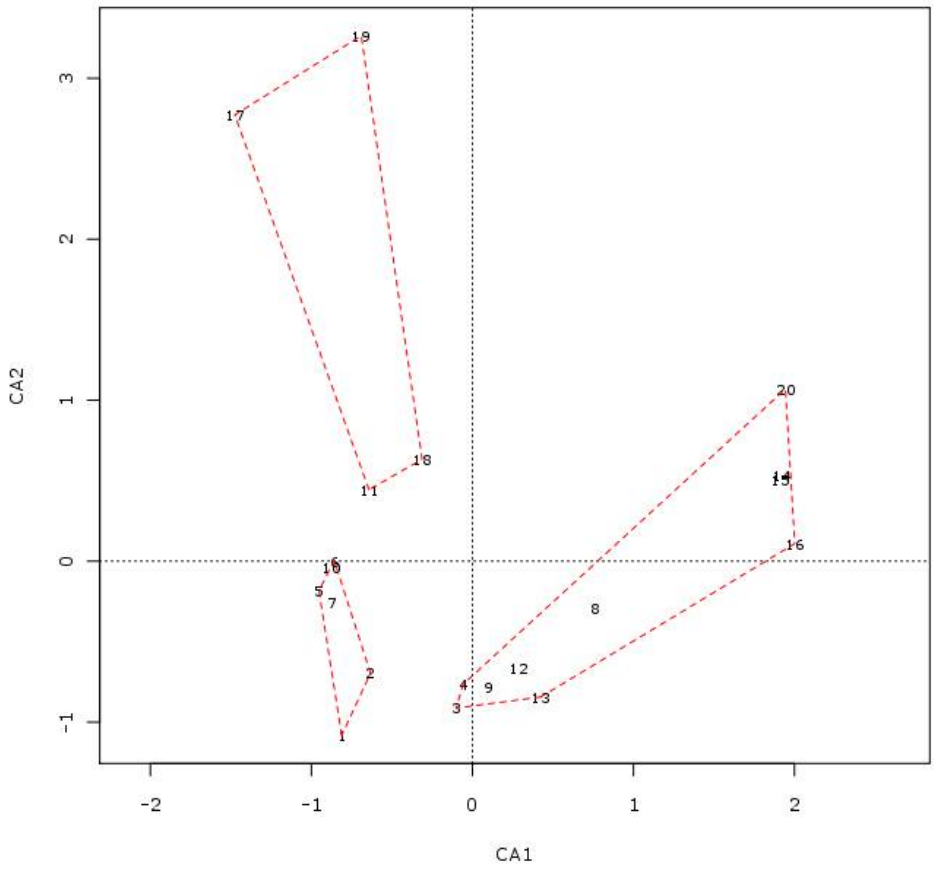

Figura 5: Ordenación con grupos previos en una clasificación

3.4. Tablas de comunidades clasificadas

El objetivo de una clasificación es muchas veces crear un cuadro de comunidades vegetales. Para esto los inventarios y las especies deben ordenarse de modo que la tabla resultante tenga un aspecto estructurado; sin embargo el cluster obtenido de una clasificación no muestra esta estructuración.

La librería base de «R» dispone de alguna función para hacer esto (**dendrogram**), pero vegan incluye una mucho más potente (**vegemite**) que produce cuadros de vegetación compactos; puede incluso trabajar con los resultados de una ordenación:

```
library(vegan)
read.table("yesos.dat", header=T)->yesos
## Se muestran sólo las especies más comunes
freq<-apply(yesos > 0,2,sum)
vegemite(yesos,[3]scale="Braun.Blanquet",sp.ind=freq>10)
## Ordenar usando un análisis de correspondencias
vegemite(yesos,use=decorana(yesos),"Braun.Blanquet",zero="-")
```

En ocasiones la transformación es para que un resultado de R pueda ser leído por otro programa. Por ejemplo, son varios los programas de análisis genético a los que el fichero de entrada debe ser una matriz originada a partir de la aplicación a los datos de una medida de similitud (típicamente el índice de **Jaccard**); sin embargo cuando R calcula este índice se crea una semimatriz en diagonal que es rechazada por las aplicaciones para genética (por ejemplo el programa **PAUP**); en este caso debe indicarsele a R que cree una matriz rectangular; el ejemplo siguiente s

3.5. Uso de R para análisis genéticos

La matriz de similitud debe ser pasada a ciertos programas de análisis genético siendo rectangular (por ejemplo el programa PAUP), pero la aplicación de índices de similitud crea semimatrices. Veamos con el fichero «yesos.dat» como conseguir el archivo con la estructura adecuada:

```
library(vegan)
read.table("yesos.dat",header=T,sep=",")-> yesos
dim(yesos)  #el resultado es [1] 29 59, interesa las 29 filas (unidades muestrales)
as.matrix(vegdist(yesos,"jaccard"))-> mimatrizdejaccard
write(mimatrizdejaccard,"jaccard.txt",ncolumns=29)
```

3.6. Conclusiones

La clasificación jerárquica puede ayudar a sacar luz de nuestros datos y su aplicación en la fitosociología es muy clara; sin embargo la interacción del investigador con el análisis es muy directa, afectando a tres pasos del mismo:

9. Medida de similitud, determinando la matriz de distancias
10. Estrategia de fusión, determinando cómo se agrupan los objetos
11. Punto de corte, determinando cuántos grupos se van a reconocer

4. Ordenación

4.1. Introducción

Las técnicas de ordenación más utilizadas son las de tipo métrico basadas en el método de los *Eigenvectores*, dentro de estas el análisis de componentes principales (**PCA**) y el análisis de correspondencias (**CA**) son los más utilizados, siguiéndoles de lejos el análisis de coordenadas principales. El PCA está basado en la *distancia euclídea*, mientras que el CA utiliza la como distancia el χ *cuadrado*, el análisis de coordenadas principales puede usar cualquier tipo de medida de distancia, pero cuando utiliza la euclídea su resultado es idéntico al del PCA.

4.2. Análisis de componentes principales (PCA)

Esta técnica rota los datos en el espacio vectorial, de tal modo que la mayor parte de las variaciones se acumula en los primeros ejes. Utiliza la distancia euclídea como elemento métrico.

Realiza una representación linear de los datos, generalmente poco apropiada para el análisis de comunidades pero resulta muy potente en la reducción de las medidas ambientales.

3 Se puede seleccionar un grupo de unidades muestrales con **select=c(57, 116,129)**

El análisis de componentes principales (PCA) puede realizarse con una función de la librería básica de «R»: **prcomp** o **princomp**. Para su adecuado funcionamiento el número de variables (especies) debe ser menor que el de unidades muestrales (inventarios); de no cumplirse esto hay que transponer la matriz:

```
read.table("yesos.dat", header=T)->yesos
t(yesos)->yesos2
biplot(princomp(yesos2))
cor(yesos2)
```

En «vegan» PCA se realiza con la función **rda**:

```
library(vegan)
data(varespec)
vare.pca<-rda(varespec)
plot(vare.pca)
```

Esta última orden nos da la siguiente salida:

```
Call:
rda(X = varespec)
              Inertia Rank
Total            1826
Unconstrained    1826   23
Inertia is variance
Eigenvalues for unconstrained axes:
   PC1    PC2    PC3    PC4    PC5    PC6    PC7    PC8
982.98 464.30 132.25  73.93  48.42  37.01  25.73  19.71
(Showed only 8 of all 23 unconstrained eigenvalues)
```

La salida nos dice que la inercia total es 1826 y esa inercia es la varianza. En otras palabras, cada eje explica una cierta proporción de varianza (el primero 983/1826 = 53.8%). Si vemos el gráfico [**plot(vare.pca)**] el resultado es poco satisfactorio, dos especies de líquenes son las únicas visibles (*Cladina, Pleurozium schreberi* y todas las demás están amontonadas cerca del origen de coordenadas. Esto es debido a la equivalencia entre inercia y varianza, pues sólo las especies con elevada abundancia tienen significado en el resultado. Estandarizar todas las especies para unificar la varianza, o usar coeficientes de correlación en vez de covarianzas dará un resultado más equilibrado:

```
vare.pca<-rda(varespec,scale = TRUE)
vare.pca
```

Y este es el resultado:

```
Call:
rda(X = varespec, scale = TRUE)
              Inertia Rank
Total              44
Unconstrained      44   23
Inertia is correlations
Eigenvalues for unconstrained axes:
  PC1   PC2   PC3   PC4   PC5   PC6   PC7   PC8
8.898 4.756 4.264 3.732 2.964 2.884 2.727 2.180
(Showed only 8 of all 23 unconstrained eigenvalues)
```

Ahora la inercia es la correlación, el total de inercia es igual al número de variables (especies). El porcentaje explicado por el primer eje decrece respecto al del PCA anterior. Podemos ver el gráfico con *plot(vare.pca, scaling = 3)*

4.3. Análisis de correspondencias (CA)

Semejante al PCA, reemplaza la métrica euclídea por la ; además presupone un modelo de respuesta unimodal de las

especies frente a los grandientes ambientales, no linear con en el anterior. Resulta un método mucho mejor que aquel para la ordenación de comunidades.

```
library(vegan)
data(varespec)
vare.ca<-cca(varespec)
plot(vare.ca)
```

Este es el resultado:

```
Call:
cca(X = varespec)
              Inertia Rank
Total           2.083
Unconstrained   2.083   23
Inertia is mean squared contingency coefficient
Eigenvalues for unconstrained axes:
    CA1     CA2     CA3     CA4     CA5     CA6     CA7     CA8
0.52493 0.35680 0.23444 0.19546 0.17762 0.12156 0.11549 0.08894
(Showed only 8 of all 23 unconstrained eigenvalues)
```

Ahora la inercia es el coeficiente de cuadrados medios de la contingencia, basada en la distancia *χ cuadrado*.

4.4. Análisis de correspondencias segmentado (DCA)

El CA es un método mucho más robusto en la ordenación de comunidades que el PCA. Sin embargo en gradientes ecológicos prolongados muestra algunos defectos que fueron corregidos en el análisis de correspondencias segmentado o DCA (detrended correspondence analysis):

- Gradientes simples pero prolongados, aparecen como curvas o arcos en la ordenación. La solución es segmentar los ejes.
- Las unidades muestrales se agrupan en los extremos del gradiente, para evitar esto habría que reescalar los ejes e igualar la varianza.
- Las especies raras influyen mucho en el resultado, es necesario quitarles peso.

Mientras en el PCA la curva tiende a ser convexa, con la parte central hacia arriba, en CA la curva suele ser cóncava.

Todos estos trucos se incorporan en la función de «**vegan**» **decorana**, cuyo funcionamiento es simple: **vare.dca <-decorana(varespec)**; podemos ver el resultado gráfico (**plot(vare.dca)** o si después queremos ver **vare.dca**, obtenemos lo siguiente:

```
Call:
decorana(veg = varespec)
Detrended correspondence analysis with 26 segments.
Rescaling of axes with 4 iterations.
                 DCA1   DCA2   DCA3    DCA4
Eigenvalues     0.5235 0.3253 0.2001 0.19176
Decorana values 0.5249 0.1572 0.0967 0.06075
Axis lengths    2.8161 2.2054 1.5465 1.64864
```

El DCA cumple tres objetivos:

1. Segmenta, para eliminar la curvatura producida por el CA.
2. Reescala, para corregir la acumulación de unidades muestrales en los extremos de los ejes de ordenación.
3. Baja el peso, para reducir la influencia de las especies raras.

Sin embargo el DCA también crea artefactos, como la tendencia a disponer los objetos en un triángulo o diamante, así como retorcer el espacio en la representación final.

Hay toda una escuela a favor de esta técnica para ordenaciones no canónicas, pero los trucos utilizados tienen escasa o nula base matemática.

4.5. Gráficos de ordenación congestionados

A veces basta con especificar en la salida gráfica que sólo se muestren las especies (species) o las unidades muestrales (sites):

```
library(vegan)
data(dune)
dune.pca<-rda(dune)
plot(dune.pca,display="sites")
```

Si esto es insuficiente se puede utilizar la función **ordiplot** (gráfico de puntos para diagramas congestionados) en vez de «spec» «sites» para poner las unidades muestrales o las especies en el diagrama.

```
library(vegan)
data(dune)
ordina<-cca(dune)
tmp<-ordiplot(ordina)
identify(tmp,"spec")
```

Señale con el ratón sobre las cruces del diagrama o cerca de ellas; verá como van apareciendo los nombres de las especies.

4.6. Análisis multivariantes constreñidos (canónicos p.p.)

En las ordenaciones no forzadas primero encontramos la mayor variación de nuestros datos y luego relacionamos esta variación con las observaciones ambientales en el campo. En el caso de las ordenaciones constreñidas sólo queremos mostrar la variación que puede ser explciada por las variables ambientales utilizadas.

«Vegan» tiene dos métodos de ordenación constreñida que en realidad son las versiones adaptadas de los dos métodos básicos de ordenación: RDA y CCA.

Vamos a teclear los siguientes comandos[4] :

```
library(vegan)
data(varespec)
data(varechem)
vare.cca<-cca(varespec ~ Al + P + K, varechem)
plot(vare.cca)
```

El último comando provoca la siguiente salida:

```
Call:
cca(formula = varespec ~ Al + P + K, data = varechem)
              Inertia Rank
Total          2.0832
Constrained    0.6441    3
Unconstrained  1.4391   20
Inertia is mean squared contingency coefficient
Eigenvalues for constrained axes:
  CCA1   CCA2   CCA3
0.3616 0.1700 0.1126
Eigenvalues for unconstrained axes:
    CA1     CA2     CA3     CA4     CA5     CA6     CA7     CA8
0.35004 0.22008 0.18507 0.15512 0.13511 0.10027 0.07730 0.05369
```

4 El símbolo se obtiene en linux pulsando la tecla «F5» o bien con la combinación «AltGr + 4».

```
(Showed only 8 of all 20 unconstrained eigenvalues)
```

La salida es similar a la de un PCA, pero ahora la inercia se descompone en componentes forzados y no forzados, hay tres componentes en los primeros, derivados de las variables utilizadas.

La salida gráfica básica se obtiene con **plot(vare.cca)**, pero una salida más espectacular la podemos tener si usamos la librería **scatterplot3d**:

```
library(scatterplot3d)
ordiplot3d(vare.cca,type="h")
```

5. Combinación de ordenación y clasificación

En esta sección se va a ver como combinar ambos métodos puede en ocasiones generar unos resultados que clarifican la interpretación de los datos.

Se trata de hacer una ordenación sobre los datos originales y aplicar a los resultados (scores) de la misma una clasificación bastante estandarizada.

```
library(vegan)
scores(cca(yesos),display="sites",choices=c(1,2))->prueba
plot(hclust(dist(prueba),"war"),hang=-1)
```

Puede hacer la prueba en vez de con «sites» con «species»

6. Creando ficheros para impresión e intercalar en documentos

Podemos desde R hacer que la salida gráfica, que por defecto es al monitor del ordenador, se realice a un fichero, destacando por su utilidad los de formato «**pdf**» y el vectorial «**svg**».

Antes de hacer la llamada a la función (por ejemplo «diagram») es preciso cargar la librería **cairoDevice** y cambiar la interfaz de salida, desde monitor a **pdf** o **svg**, dando el nombre del fichero, y después de lanzar la función hay que cerrar la salida para que de nuevo esta se redirija a la pantalla.

Vamos a ver un ejemplo para ambas salidas, suponiendo que queremos el diagrama climático de la estación de «tamatave»:

- Salida a pdf

```
library(cairoDevice)
source("diagram.R")
Cairo(file="tamatavediagrama.pdf",width=7,height=7,pointsize=10)
diagram("tamatave")
dev.off()
```

Basta cambiar la extensión del nombre del fichero a «ps», «png», «svg» para obtener los ficheros en los formatos correspondientes. Al final de estos procesos tendremos en el directorio de trabajo los ficheros con el nombre que hemos puesto y la extensión correspondiente, los cuales podrá intercalar sin problemas en un documento.

7. Fuentes de consulta

7.1. Documentos

Venables, W.N. y Smith, D.M. 2003. *An introduction to R. Notes on R: A Programming Environment for Data Analysis and Graphics. Version 1.6.2*. Documento PDF.
Oksanen, J. 12/01/2007. *Multivariate Analysis of Ecological Communities in R: vegan tutorial*. Documento PDF.
Venables, W.N. y Smith, D.M. 2003. *The R Reference index*. Documento PDF.

7.2. Páginas de Internet

http://cc.oulu.fi/~jarioksa/softhelp/vegan.html
http://gilgames.bio.um.es/r.txt
http://labdsv.nr.usu.edu/splus_R/
http://www.ci.tuwien.ac.at/R/