IMM con R. Lab6: Análisis de Coordenadas Principales

Dra. Maite Mascaro y Dr. Edlin Guerra Castro

27/03/2022

Caso Ekofisk macrofauna y contaminantes en sedimento

Volvemos al estudio publicado por Gray JS, Clarke KR, Warwick RM, Hobbs G (1990). Detection of initial effects of pollution on marine benthos: an example from the Ekofisk and Eldfisk oilfields, North Sea. Marine Ecology Progress Series 66, 285-299.

Este estudio consiste en la evaluación del macrobentos y varios contaminantes del sedimento en 39 sitios dispuestos en un diseño radial alrededor de una plataforma de perforación petrolera en el mar del Norte, donde se espera que los contaminantes asociados a la actividad petrolera afecten la estructura del ecosistema. La disposición de los sitios es circular, alejándose cada ciertos kilómetros del centro de perforación. A diferencia de la actividad 4, ahora sí consideraremos las distancias de cada sitio respecto a la plataforma. Usaremos dos métodos de ordenación: PCO y MDS para representar las matriz de la macrofauna.

1. Construcción de un PCO con álgebra de matrices

a) Importe el archivo "macrofauna.cvs" y renómbrela `datos`. Identifique la información de sus columnas y filas.

b) Recuerde que se deben filtrar las primeras dos columnas. Llame a la nueva matriz "macrofauna"

```
macrofauna <- datos[,3:length(datos)]
```

c) Aplique el índice de similitud Bray-Curtis luego de transformar las abundancias a raíz cuarta. Llame a la matriz de abundancias transformadas `r.c` y a la matriz Bray-Curtis resultante `bray1`.

```
r.c <- macrofauna^(1/4)
bray1 <- vegdist(r.c)
```

d) Genere un PcoA de bray1, usando la álgebra matricial vista en clase. El primer paso es convertir a `bray1` en una matriz cuadrada.

```
bray1.m <- as.matrix(bray1)

# Luego, genere la matriz A
A<- -0.5*(bray1.m^2)
dim(A)

# Seguidamente, se debe centrar A en sus propias filas y columnas para
# generar la matriz Gower G. Trate de describir qué se está efectuando en cada paso.
nn <- dim(A)[1]
nn
ones <- matrix(1,nn)
ones
I <- diag(1,nn)
I
```

```
dim(I)

Gower <- (I-1/nn*ones %*% (t(ones))) %*% A %*% (I-1/nn*ones %*% (t(ones)))
dim(Gower)
```

Estimada la matriz de Gower, se descomponen los autovalores para obtener los ejes o *scores* usando la función `eigen`

```
EG <- eigen(Gower)
```

Para obtener la ordenación del PCoA, genere las coordenadas principales usando los autovalores con el script:

```
#Estimación de coordenadas
vectors <- sweep(EG$vectors, 2, sqrt(abs(EG$values)), FUN = "*")

# Identificamos a las filas de "vectors" para proyetar las etiquetas en el gráfico
row.names(vectors) <- paste(row.names(r.c))

# Generamos la ordenación usando la función ordiplot{vegan}.
# Genere el PCO proyectando las etiquetas, y luego otro PCO proyectando puntos.
```

También podríamos generar la ordenación del PCO usando `ggplot2`. Esto sería como iniciar de cero, en términos de generación del gráfico, pero permitiría proyectar el PCO con las características gráficas que más nos gusten. Primero se debe generar una data.frame con los vectores con información real (excluyendo los vectores imaginarios)

```
scoresPCO <- as.data.frame(vectors)

# Luego le incluímos al data.frame creado el factor distancia.
# Esto será útil para proyectar un gráfico cuyos puntos estén diferenciados
# por el factor distancia.

scoresPCO$dist <- datos$`#dist`

distancia <-
  as.factor(scoresPCO$dist) #con esto definimos un vector de clase "factor"

PCO.ggplot <-
  ggplot(scoresPCO, aes(x = V1, y = V2, colour = distancia)) +
  geom_point(size = 3) +
  ylab(expression(paste("PCO2"))) +
  xlab(expression(atop(paste("PCO1"))))# Proyección del PCO

PCO.ggplot #imprimir gráfico en la consola
```

2. Construcción de un PCO con las funciones "enlatadas" cmdscale{stats}, pcoa{ape}, ordiplot{vegan} y biplot {stats}.

a) Aplique las funciones cmdscale sobre la matriz bray1. ¿Se obtiene la misma ordenación que aplicando manualmente el álgebra de matrices? Compare los Scores y el gráfico.

```
PCO.r <- cmdscale(bray1, k = (nrow(r.c) - 1), eig = TRUE)
ordiplot(scores(PCO.r)[, c(1, 2)], type = "p", main = "PCoA con cmdscale y Vegan")
ordiplot(scores(PCO.r)[, c(1, 2)], type = "t", main = "PCoA con cmdscale y Vegan")
```

b) Genere otro PCO pero usando la función `pco` del paquete `ape`.

```
PCO.ape <- pcoa(bray1)
```

c) Ahora tratemos de hacer un biplot, proyectando las especies responsables. USe la función `biplot` para tal fin

```
biplot(
  PCO.ape,
  Y = r.c,
  plot.axes = c(1, 2),
  dir.axis1 = 1,
  dir.axis2 = 1,
  main = "PCO pcoa{ape} 2- Macrofauna"
)
```

d) Mejoremos el PCO incluyendo solo a las 10 especies que mejor contribuyen a diferenciar los grupos en las distancias 1 y 4. Para esto usaremos la herramienta SIMPER desarrollada por Clarke (1993), original del software PRIMER, pero incluída en el paquete `vegan`. Esta rutina descompone las similitudes entres cada par de sitios y relativiza el peso de cada especie en contribuir a la disimilitud promedio entre dos grupos.

e) Apliquemos *simper* a la matriz "r.c" usando el factor distancia. Para poder acceder al resultado, llamaremos a este `simper_1v4` y al resumen de los resultados `sum1v4`

```
simper_1v4 <- simper(r.c, distancia)
sum1v4 <- summary(simper_1v4)
```

f) El objeto "sum1v4" es una lista, y como tal, podemos observar su elementos uno a uno. Identifiquemos las especies que mayor contribución a las diferencias entre los grupos 1 y 4 con el siguiente comando

```
sum1v4$`1_4`
```

g) Generemos un vector de nombre "sp" con los nombres de las 20 especies con mayor contribución a las diferencias entre la distancia 1 y 4 y usémoslo como subsetting en la matriz "r.c", que simultáneamente podemos convertir de data.frame a matriz con el nombre "r.c.sub"

```
sp <- rownames(as.data.frame(sum1v4$`1_4`[1:20, ]))
r.c.sub <- as.matrix(r.c[, sp])
```

h) Repitamos el biplot pero con la subselección de especies definidas en sp

```
biplot(PCO.ape, Y=r.c[,sp],
       plot.axes = c(1,2),
       dir.axis1=1,
       dir.axis2=1,
       main="PCO pcoa{ape} 2- Macrofauna")
```

i) Ya vimos varias de las alternativas para proyectar nuestro PCO, todas mejorables desde el punto de vista estético, para ello hay que ver los atributos de las funciones que decidamos utilizar. Lo más importante de todo es identificar si el PCO es una buena representación gráfica. ¿Cómo hacemos eso? Una forma (no laúnica) es explorando al objeto PCO.ape.

```
PCO.ape$values$Relative_eig[1:2] # Autovalores del PCO1 y PCO2

PCO.ape$values$Relative_eig[1:2] * 100 # Valores convertidos %

sum(PCO.ape$values$Relative_eig[1:2] * 100)
#esta suma indica cuánta de la variación en la matriz bray1 es proyectada por los dos
#primeros ejes del PCO. ¿cree usted que es una buena ordenación?
```

3. Ejecutemos ahora un MDS con la función metaMDS{vegan} y isoMDS{MASS}

a) Inspeccione el archivo de ayuda de metaMDS, y ejecute el escalamiento. Acá se proponen estos scripts, pero usted puede mejorar el gráfico notablemente explorando las opciones gráficas de las función `plot`

```
MDS <- metaMDS(r.c,
               distance = "bray",
               k = 2,
               trymax = 30)
plot(MDS, display = "sites")
text(MDS, display = "sites")
title(main = "MDS con metaMDS, Bray-Curtis macrofauna")
mtext(text = paste("stress =", round(MDS$stress, 3)), outer = FALSE)
```

b) Ahora inspeccione el archivo de ayuda de isoMDS y ejecute el escalamiento.

```
MDS2 <- isoMDS(bray1)
plot(MDS2$points)
shp <- Shepard(d = bray1, x = MDS2$points)
plot(shp)
lines(shp$x, shp$y, type = "S")
```

c) Usemos a MDS2 para generar un MDS con puntos identificados según el factor distancia

```
# Extraer coordenadas de cada muestra
MDS2.ejes <- as.data.frame(MDS2$points)

#Agregar el factor distancia a la tabla
MDS2.ejes$dist <- distancia

#Asegurar márgenes amplios para el gráfico (opcional)
par(mar = c(3, 2, 3, 2))

#Ordenación base
plot(
  MDS2.ejes[, 2] ~ MDS2.ejes[, 1],
  asp = 1,
  cex = 1,
  xlab = "MDS1",
  ylab = "MDS2"
)

#Modificar colores de puntos según distancia
with(subset(MDS2.ejes, dist == 1), points(V1, V2, pch = 21, bg = "red"))
with(subset(MDS2.ejes, dist == 2), points(V1, V2, pch = 21, bg = "yellow"))
with(subset(MDS2.ejes, dist == 3), points(V1, V2, pch = 21, bg = "green"))
with(subset(MDS2.ejes, dist == 4), points(V1, V2, pch = 21, bg = "blue"))

#Título del gráfico
title(main = "MDS con isoMDS, Bray-Curtis macrofauna")

#Agregar el valor del stress al gráfico
mtext(text = paste("stress =", round((MDS2$stress/100),3)), outer = FALSE)
```

d) Compare todas las ordenaciones que ha generado de bray1, tanto los PCO, como los biplot y los MDS. ¿Cuál le parece mejor?

4. El PCO no fue lo suficientemente flexible como para refrejar a bray1. Sin embargo, es una herramienta muy útil para poder estimar "semi" parámetros, como los centroides de los grupos. Usemos los scores generados con el análisis de matrices para estimar los centroides y proyectarlos.

a) Usemos el paquete `dplyr` para manejar con mayor facilidad la data.frame scoresPCO. Lo primero es convertir esa data.frame a tabla con la función `as.tbl`

```
scoresPCO.tb <- as_tibble(scoresPCO)
```

b) Luego, se debe indintificar al factor dist con la función "group_by". Esa identificación debe tener un nombre particular. Usemos el nombre scoresPCO.dist

```
scoresPCO.dist <- group_by(scoresPCO.tb, dist)
```

c) Generemos los promedios para todos los scores usando como agrupación al factor distancia. Llame a la salida `centroides`. Luego de aplicar el script trate de identificar las dimensiones de `centroides`. ¿Reconoce lo que se calculó?

```
centroides <- scoresPCO.dist %>%
                summarise_all(mean) %>%
                 as.data.frame()

centroides
```

d) Calcule las distancias euclideanas a centroides y llame a la salida `dist.cen`. Proyecte esta matríz de distancias con un PCO usando `cmdscale`

```
dist.cen <- vegdist(centroides[2:33], method = "euclidean")
PCO.cen <- (cmdscale(dist.cen))
ordiplot(PCO.cen)
```

e) ¿qué puede concluir respecto las diferencias en la macrofauna según la distancia según TODOS los métodos que hemos empleado (aproximación univariada, clusters, PCO, MDS, distancia entre centroides)