Análisis Multivariado Aplicado a Ciencias Ambientales usando R

Actividad 3

Prof. Edlin Guerra Castro

June 20, 2024

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 anterior, ahora sí consideraremos los grupos de distancias de cada sitio respecto a la plataforma. Usaremos tres métodos de ordenación: PCA, PCO y MDS.

(a) Diseño de muestreo radial (b) Plataforma Ekofisk (c) Mar del Norte

Figura 1. Plataforma de exploración petrolera Ekofisk

Parte 1: Análisis de Componentes Principales (PCA)

Siga las siguientes indicaciones:

a) Importe la matriz de datos "sedimentos".

b) Explora los datos utilizando herramientas gráficas y numéricas apropiadas. Recuerda que puedes usar subsetting para seleccionar solo las columnas o filas de la tabla que contengan las variables que te interesan. (PISTA: recuerda funciones como `summary`, `plot` y `pairs`).

c) De la exploración aprendiste que las escalas de las variables difieren mucho. Para evaluar el problema de la diferencia en escalas entre variables, generemos dos análisis, una con escalas originales y otra con datos con transformaciones individuales y normalizados. Para el análisis con datos en escala original, obtén la matriz de varianza-covarianza a `sed` mediante la función `cov` y llámala `C`. ¿Qué dimensiones tiene la matriz C?

```
C <- cov(sed)
```

d) ¿Qué representan los elementos de la diagonal? ¿Qué representan los elementos fuera de la diagonal? Confirma esto usando tu conocimiento en estadística básica? ¿Qué representa la traza (suma de elementos de la diagonal) de esta matriz en el juego de datos?

e) El siguiente código es para obtener la matriz varianza-covarianza a mano. A cada código lo anteceden los comentarios para ayudarte a seguir las instrucciones que están en la presentación de la clase. Corre el código paso por paso, y compara el resultado con el obtenido con la función `cov`.

```
#Debemos garantizar que sed sea una matriz, cambiando nombre a Y
Y <- as.matrix(sed)

#Obtención de un vector vertical unitario
ones <- matrix(1, nrow = nrow(Y), ncol = 1)
dim(ones)

#Cálculo del número de filas (número de observaciones)
numFilas <- t(ones) %*% ones
numFilas

#Obtención del inverso de esa matriz de un único elemento
invnumFilas <- solve(t(ones) %*% ones)
invnumFilas

#Obtención de la sumatoria de elementos por columnas
```

```r
sumaCol <- t(ones) %*% Y
sumaCol

#Mutiplica la sumaCol por el inverso del numFilas (divide la sumatoria entre n filas).
invnumFilas %*% sumaCol

#El vector de medias también se puede obtener así.
colMeans(Y)

#Multiplica el vector horizontal de medias por el vector vertical unitario para obtener
#una matriz de medias.
Y.bar <- ones %*% invnumFilas %*% sumaCol
dim(Y.bar)

#Sustraer la matriz de datos de la matriz de medias.
Z <- Y - Y.bar

#Esto también se logra obteniendo los residuales de una regresion lineal contra solo un
#intercepto de valor 1.
Z <- resid(lm(Y ~ 1))

#Obtención de la matriz de suma de cuadrados y cross-products (SSCP matrix) (elevar al
# cuadrado la matriz Z)
S <- t(Z) %*% Z
S

#SSCP es una matriz triangular con la SS en la diagonal y los CP en los elementos fuera
#de la diagonal.
dim(S)

#Obtener la matriz de varianza covarianza: Dividir SSCP (suma de cuadrados y productos
#cruzados) entre n-1 (g.l.)
Cmano <- 1 / (nrow(Y) - 1) * S
Cmano
C
```

f) Aplica un eigenanálisis a la matriz `C` para obtener los eigenvectores y eigenvalores de un PCA (de covarianza). Usa la función `eigen`.

```r
# Estimación de autovalores y autovectores
pca.C <- eigen(C)
```

g) ¿Qué porcentaje de la variación total está explicada por el primer componente? ¿Cuánta por el segundo? ¿Cuál es la variable con la carga positiva más alta en el segundo componente? ¿Qué significa eso? ¿Con qué componentes principales te parece que queda explicada suficiente variación en este estudio?

```
#Autovalores
pca.C$values

#Autovalores relativos respecto a la variación total
round(pca.C[["values"]]/sum(pca.C[["values"]]),3)

#autovectora
pca.C$vectors
```

h) Procedamos a generar la ordenación en 2D de un PCA, usando los primeros componentes.

```
#Ajudica el nombre E.cov a una matriz con los eigenvectores de un PCA de covarianza
E.cov <- pca.C$vectors

#Obtiene las proyecciones como el producto matricial de Z (matriz de residuales Y-Y.barra)
#y la matriz E.cov
P.cov <- Z %*% E.cov

plot(P.cov[,2]~P.cov[,1],asp=1,cex=1, xlab = "PC1", ylab = "PC2")
```

i) Generemos ahora el PCA de covarianza con la función `prcomp` sobre la matriz Y, usando el siguiente código. Compara con el PCA a mano.

```
pca.c <- prcomp(sed,
                  retx = TRUE,
                  center = TRUE,
                  scale. = FALSE)

ordiplot(pca.c)
```

j) Hemos comprobado que la función incorporada en R para hacer PCA y los procedimientos manuales generaron el mismo resultado. No obstante, es evidente que el problema de escalas generó una ordenación que no debe ser interpreta. Construya ahora PCA basado en correlaciones. Para ello, aplique primero el pretratamiento (calcule el logaritmo) a las variables que no tengan distribución "simétrica". Seguidamente resuelva el problema de las diferencias en unidades y escalas de las variables normalizando los datos (centrando en cero y valores reflejados en varianza) usando la función `decostand` de `vegan` con el

argumento `method` adecuado ("standardize") y llame a la matriz resultante "sed.stand". Luego aplique la función `prcomp` (note que podría directamente pasar de `sed1` a `prcomp` definiendo los argumentos center y scale, con TRUE).

```
# Transformaciones individuales
sed1<-data.frame("log.THC"=log(sed$THC),
"log.Cu" = log(sed$Cu),
"log.Ba" = log(sed$Ba),
"log.Pb" = log(sed$Pb),
"log.Ni" = log(sed$Ni),
"log.Sr" = log(sed$Sr),
sed[2:4])

# Normalización (media 0 y varianza 1)
sed.stand<-decostand(sed1, "standardize")

#construcción del PCA con la función prcomp
pca.cor <- prcomp(sed1,
                  retx = T,
                  center = TRUE,
                  scale. = TRUE)

pca.cor

#Generación del gráfico
ordiplot(pca.cor)
```

k) ¿Qué porcentaje de la variación total está explicada por el primer componente? ¿Cuánta por el segundo? ¿Cuál es la variable con la carga positiva más alta en el primer componente? ¿Qué significa eso?

l) Podemos ahora proyectar los autovalores usando la funcion `biplot`. Esto nos ayudará en enternder qué variables son responsables del patrón observado:

```
biplot(
  princomp(sed1, cor = T),
  choices = 1:2,
  scale = 0,
  var.axes = T,
  arrow.len = 0.1,
  col = c("black", "red"),
  cex = 0.7,
  asp = 1,
```

```
  xlab = "PC1",
  ylab = "PC2"
)
```

10. Otras visualizaciones: Explore el paquete `ggfortify` y genere ordenaciones basados en PCA visualmente amenos.

```
library(dplyr)
library(ggplot2)
library(ggfortify)

distancia <- sedimentos|>
  mutate(distancia = if_else(Distance <= 0.25, "<250m",
                             if_else(Distance <= 1.0, "<1.0 km",
                                     if_else(Distance < 3.5, "< 3.5 km","> 3.5 km"))))|>
  relocate(distancia, .after = Distance)|>
  mutate(distancia = factor(distancia, levels = c("<250m", "<1.0 km", "< 3.5 km","> 3.5 km")))

autoplot(pca.cor, data = distancia, colour = 'distancia',
         loadings = TRUE, loadings.colour = 'blue',
         loadings.label = TRUE, loadings.label.size = 3)
```

Parte 2. Análisis de Coordenadas Principales (PCO)

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"

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`.

d) Genere un PCO 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.

ordiplot(vectors)
```

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)

```r
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$distancia <- as.factor(datos$`#dist`)

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

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

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.

```r
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`.

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

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

```r
biplot(
  PCO.ape,
  Y = r.c.sub,
  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`

```
distancia <- as.factor(datos$'#dist')
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 la única) es explorando al objeto PCO.ape.

```r
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?
```

Parte 3. Escalamiento Muldimensional (MDS)

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

```r
nMDS <- metaMDS(r.c,
                distance = "bray",
                k = 2,
                trymax = 50)
plot(nMDS, display = "sites")
title(main = "MDS con metaMDS, Bray-Curtis macrofauna")
mtext(text = paste("stress =", round(nMDS$stress, 3)), outer = FALSE)

# Gráfico de Shepard
shp1 <- Shepard(d = bray1, x = nMDS$points)
plot(shp1)
lines(shp1$x, shp1$yf, type = "S")
```

b) Ahora inspeccione el archivo de ayuda de `monoMDS` y ejecute un escalamiento métrico usando regresión lineal.

```r
library(MASS)

# Generación de un MDS con stress calculado usando un modelo lineal
mMDS <- monoMDS(bray1, model = "linear", maxit = 1000)
plot(mMDS$points)

# Gráfico de Shepard
shp2 <- Shepard(d = bray1, x = mMDS$points)
plot(shp2)

# Identificación de parámetros para ajuste lineal
```

```
lm(y~x, data = shp2)

plot(shp2)
abline(a = -1.995, b = 8.13)
```

c) Usemos a nMDS y `ggplot` para generar un MDS con puntos identificados según el factor distancia

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

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

#Gráfico base
nMDS.plot <- ggplot(nMDS.ejes, aes(x = MDS1, y = MDS2, colour = distancia))+
  geom_point(size = 4)

#Mejora estética del nMDS
nMDS.plot +
  theme_bw()+
    theme(
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank()
  )+
  theme(axis.ticks = element_blank(), axis.text = element_blank())+
  ggtitle("nMDS con monoMDS, Bray-Curtis macrofauna",
          subtitle = paste("stress =", round((lMDS$stress),3)))
```

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

Visualización de centroides

El PCO no fue lo suficientemente flexible como para reflejar 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`

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

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

```
library(dplyr)
scoresPCO.dist <- group_by(scoresPCO.tb, distancia)
```

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, type = "text")
```

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)