Propiedades de la poblaciÃ³n

Carlos IvÃ¡n Espinosa

4 de octubre de 2019

Contents

Introducción **1**

Distribución geográfica **2**

Preparación de datos 2

Datos de ausencia y de fondo 7

Datos ambientales 9

Ajuste del modelo, predicción y evaluación 11

Métodos de modelamiento 14

Combinando predicciones del modelo 23

Introducción

Nosotros como individuos que tan consientes somos de pertenecer a un grupo más grande de individuos, una población, y que de una u otra manera las características de ese grupo tiene un efecto sobre lo que nos pasa a nosotros individualmente. Pensemos en una propiedad del individuo la cantidad de hijos, es igual la cantidad de hijos que ahora tienen las parejas en relación a lo que tenían los abuelos de estas. Al igual que la población humana, las poblaciones de otros organismos tienen efectos sobre los organismos y esos efectos a su vez tienen un impacto sobre las propiedades de la población. Durante los últimos años los cambios que se están suscitando alrededor del planeta están afectando a las poblaciones de diversas especies. Poder entender como una población responde a esas presiones es fundamental si queremos emprender acciones de conservación. Este análisis pasa por la posibilidad de describir eficientemente la población; ¿Dónde se puede establecer la población? ¿Cuál es la abundancia de esa población bajo determinadas condiciones? En el presente ejercicio trabajaremos en comprender las herramientas existentes para realizar la caracterización de la comunidad y sobre todo en desarrollar habilidades para realizar una interpretación biológica de esa caracterización. Una primera restricción a la cual se enfrentan los organismos es el ambiente. Las condiciones ambientales restringe donde una población puede establecerse. La capacidad de adaptación al ambiente que tienen las diferentes especies define los rangos de distribución, hay algunos animales que tienen rangos amplios de distribución y otros que tienen unos rangos restringidos. De esta forma el ambiente determina en primera medida la ***distribución geográfica*** de una especie, su *nicho fundamental*. Una vez la población se ha establecido, una propiedad importante es su tamaño poblacional, la ***abundancia poblacional***. La abundancia de la población está definida por dos factores, por un lado, el rango de distribución y por el otro la *densidad de la población*. La distribución en el espacio de la población no es necesariamente homogéneo y este depende de la escala a la cual estemos midiendo. De esta forma, las poblaciones ocupan el espacio generando un ***patrón espacial***, que a su vez determina la densidad y la abundancia.

Distribución geográfica

Como mencionamos anteriormente la distribución geográfica de una población está definida en primera medida por las condiciones ambientales, las especies ocurren ahí donde las condiciones macro-climáticas le permiten estar. Sin embargo, las especies no necesariamente se encuentran ocupando todo ese espacio, otros factores pueden limitar su distribución. Las barreras geográficas, por ejemplo, limitan las áreas que estas especies pueden realmente ocupar. Adicionalmente, las especies pueden ser excluidas de sitios ambientalmente adecuados por competencia con otras especies.

Durante los últimos años se han desarrollado múltiples modelos de distribución espacial (SDM por sus siglas en ingles) que permiten a partir de unos datos de ocurrencia generar modelos de distribución espacial. A continuación les presento una serie de pasos que permitirá desarrollar modelos de distribución espacial. El presente ejercicio está basado en el Blog *Spatial Data Science* de Robert Hijmans

Preparación de datos

Una de las fases primordiales para el desarrollo de un *SDM* son los datos de partida. Para poder implementar un modelo necesitamos obtener datos de ocurrencia (coordenadas espaciales), donde la especie ha sido catalogada. Existen múltiples bases de datos con *información de ocurrencia*, pero también los investigadores podrían tener puntos de ocurrencia levantados por ellos mismos. Una vez que se obtienen los datos, es necesario verificar que las coordenadas no sean erradas o que existan problemas con estos datos de ocurrencia, necesitamos *limpiar los datos*.

Importando datos de ocurrencia

Cuando el investigador cuenta con datos de ocurrencia levantados por él o que ha levantado de diversas fuentes, necesitamos leerlo en R. Vamos a utilizar datos de un murciélago vampiro que ocurre en las áreas tropicales y que ha sido bastante conocido por atacar al ganado. El nombre científico de esta especie es *Desmodus rotundus*.

```
library(readxl)

desmod <- read_excel("Ocurrencia Desmodus.xlsx")
```

Ahora verificamos que los datos hayan sido leídos adecuadamente.

```
str(desmod)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    251 obs. of  3 variables:
##  $ lat: num  -4.63 -4.56 -4.56 -4.51 -4.48 ...
##  $ lon: num  -79.5 -79.5 -79.5 -79.5 -80.4 ...
##  $ ID : num  170591 170114 170114 170921 168803 ...
```

La función *str* nos permite ver las propiedades de la matriz de datos leída. Como vemos tenemos 251 registros y tres variables, latitud, longitud y un identificador.

Ahora es posible que estemos iniciando el trabajo de modelado por lo que no disponemos de datos de ocurrencias, podríamos cargar los datos que han sido depositados en el Global Biodiversity Inventory Facility (GBIF) usando la función *gbif* del paquete **dismo**. Si es la primera vez que usará este paquete seguramente tendrá que descargarlo; use la siguiente línea de código para hacerlo: `install.packages("dismo")`

```
library(dismo)
```

```
## Loading required package: raster
```

```
## Loading required package: sp
```

```
# Usar las dos líneas de código a continuación para extraer
#los datos de GBIF y guardarlos en su computador

#La siguiente línea extrae los datos
# desGbif <- gbif("Desmodus", "rotundus", geo = TRUE)

#La siguiente línea graba los datos en su disco duro
# write.csv(desGbif, "desGbif.csv")

#La siguiente vez que realice el trabajo, ejecute solo leer los datos del disco duro

desGbif <- read.csv("desGbif.csv")
```

Pueden ver otros argumentos posibles a usarse con la función GBIF, se podría usar un shape o un polígono para obtener los datos de una región que no interese. A continuación generamos un mapa con el fin de verificar que los puntos tengan una ubicación correcta.

> **Nota:** se puede utilizar la función *getData* para obtener un Shapefile de un lugar que nos interese por ejemplo Ecuador o la función *shapefile* si queremos cargar un Shape que ya tengamos en el computador. Estas dos funciones se encuentran en el paquete **raster**

```
library(maptools)
data(wrld_simpl)
#graficamos un mapa del mundo
plot(wrld_simpl, xlim=c(-100,40), ylim=c(-60,40), axes=TRUE, col="light yellow")
box()
# Adicionamos los puntos de ocurrencia
points(desGbif$lon, desGbif$lat, bg='orange', pch=21, cex=0.75)
# plot points again to add a border, for better visibility
points(desmod$lon, desmod$lat, col='red', cex=0.75)
```

Limpieza de datos

Un paso fundamental para poder construir los modelos es realizar una limpieza de datos, sobre todo si los datos provienen directamente de bases de datos como la de GBIF. Como podemos observar en el gráfico anterior existe un punto en medio del Atlántico. Es necesario limpiar esta serie de inconsistencias.

Un punto importante es evaluar si no existen datos con coordenadas iguales, ver si existen datos duplicados de latitud y longitud

```
dupl <- duplicated(desGbif[, c('lon', 'lat')])
#Cuantos datos duplicados tenemos
sum(dupl)
```

```
## [1] 16955
```

```
#Tenemos 16951 datos duplicados que deberíamos excluir
#Usamos este vector para eliminar los duplicados
desGbifCd <- desGbif[!dupl,]
```

Ahora necesitamos eliminar todos los registros que no tienen datos de latitud y longitud.

```
lolNA <- which(is.na(desGbifCd$lon)&is.na(desGbifCd$lat))
desGbifCd <- desGbifCd[-lolNA,]
```

Hemos realizado algunos procesos de limpieza de datos, finalmente podemos hacer una corrección cruzada, usaremos los datos de los países y veremos si los datos corresponden a los países.

```
library(sp)
csDes <- desGbifCd
csDes$country <- droplevels(csDes$country, exclude = 0)
coordinates(csDes) <-  ~lon+lat
crs(csDes) <- crs(wrld_simpl)
class(csDes)
```

```
## [1] "SpatialPointsDataFrame"
## attr(,"package")
## [1] "sp"
```

```
ovr <- over(csDes, wrld_simpl)
head(ovr)
```

```
##   FIPS ISO2 ISO3  UN       NAME   AREA   POP2005 REGION SUBREGION      LON
## 1   MX   MX  MEX 484     Mexico 190869 104266392     19        13 -102.535
## 2   CO   CO  COL 170   Colombia 103870   4494579     19         5  -73.076
## 3   MX   MX  MEX 484     Mexico 190869 104266392     19        13 -102.535
## 4   CS   CR  CRI 188 Costa Rica   5106   4327228     19        13  -83.946
## 5   PE   PE  PER 604       Peru 128000  27274266     19         5  -75.552
## 6   MX   MX  MEX 484     Mexico 190869 104266392     19        13 -102.535
##      LAT
## 1 23.951
## 2  3.900
## 3 23.951
## 4  9.971
## 5 -9.326
## 6 23.951
```

Vemos algunos datos que tienen NA como país, estos corresponden a los datos que se encuentran en el océano y que no corresponden a ocurrencia de la especie.

```
cntr <- ovr$NAME

#which nos permite ver la ubicación de los puntos
i <- which(is.na(cntr))
length(i) #Cuantos datos sin país
```

```
## [1] 117
```

```
#Definimos las ocurrencias que coinciden
j <- which(as.character(cntr)==as.character(csDes$country))

#vemos a que corresponden los datos que no coinciden
cbind.data.frame(cntr, csDes$country)[-c(j,i),]
```

```
##               cntr csDes$country
## 39       Venezuela      Colombia
## 124      Venezuela      Colombia
## 250       Paraguay     Argentina
## 269         Brazil      Suriname
## 296     Costa Rica        Panama
## 320         Guyana      Suriname
## 336       Suriname French Guiana
## 350      Venezuela      Colombia
## 355         Guyana      Suriname
## 358         Guyana      Suriname
## 362         Guyana      Suriname
## 515         Mexico     Guatemala
## 694      Guatemala        Mexico
## 885       Suriname        Guyana
## 886       Suriname        Guyana
## 887       Suriname        Guyana
## 941         Belize        Mexico
## 951      Guatemala        Mexico
## 974      Guatemala        Mexico
## 1219        Belize        Mexico
## 1226     Guatemala        Mexico
## 1335        Brazil          Peru
## 1361          Peru       Ecuador
## 1417        Mexico        Belize
## 1552      Honduras   El Salvador
## 2012     Venezuela      Colombia
## 2133      Colombia     Venezuela
## 2922      Viet Nam      Colombia
## 2945     Venezuela      Honduras
## 2965       Bolivia        Brazil
## 2998      Paraguay     Argentina
## 3112 United States        Mexico
## 3172       Bolivia     Argentina
## 3214      Paraguay     Argentina
```

La mayoría de los datos corresponden a efectos de la precisión de los mapas, como vemos los errores se dan entre países vecinos. Si tuviésemos pocos puntos deberíamos verificar y buscar el error para no perder estos datos, sin embargo, como tenemos suficientes datos vamos a eliminar estos puntos. Eliminemos los datos que caen en el mar (NA) y los que tienen coordenadas erradas y veamos cómo quedan nuestros datos.

```
#removemos los datos que con ubicación NA
desmodF <- desGbifCd[-i,]
```

Finalmente, hemos eliminado las ocurrencias con problemas. Pero en el mapa aún tenemos puntos fuera del neotrópico, optaremos por eliminar estos puntos ya que se supone esta especie está restringida al neotrópico.

```
#Ubicamos el país
us <- which(desmodF$country=="United States")

#Registros sobre 100 grados de longitud
sa <- which(desmodF$lon>80)

desmodF <- desmodF[-c(us,sa), ]
```

Tenemos varios datos que hemos ido limpiando a lo largo de este ejercicio hasta llegar a tener unas coordenadas limpias. Sin embargo, puesto que realizaremos modelos predictivos con estas ocurrencias, es necesario reducir los posibles efectos de colinearidad espacial. Si tenemos dos registros de ocurrencia cercanos, por debajo de 500 metros (que es la resolución de las variables explicativas que veremos más adelante), es necesario quedarnos con uno solo de estos registros. A continuación vamos a eliminar los registros que se encuentran dentro de una misma celda.

```
library(dismo)
# Obtenemos los datos del paquete dismo
files <- list.files(path=paste(system.file(package="dismo"), '/ex',
                        sep=''),  pattern='grd',  full.names=TRUE )

#Usamos el primer raster para obtener la mascara
mask <- raster(files[1])
maskU <- mask
values(maskU) <- 1:ncell(mask)

#convertimos en un objeto tipo SpatialPointDataFrame
desRep <- desmodF
coordinates(desRep) <- ~lon+lat
projection(desRep) <- CRS('+proj=longlat +datum=WGS84')

#Usamos extrac para obtener el valor de cada punto de ocurrencia
xp <- extract(mask, desRep)
length(xp) #la cantidad de puntos que tenemos
```

```
## [1] 3220
```

```
length(unique(xp)) #la cantidad de puntos con una ubicación mayor a
```

```
## [1] 170
```

```
                    #500 metros
#definimos la ubicación de los duplicados
dupDes <- which(duplicated(xp))

#usamos este vector para eliminar duplicados
desmodFD <- desmodF[-dupDes,]

#graficamos nuevamente
plot(wrld_simpl, xlim=c(-100,50), ylim=c(-50,30), axes=TRUE, col="light yellow")
box()
# incluimos los puntos
points(desmodFD$lon, desmodFD$lat, bg='orange', pch=21, cex=0.75)
```

Ahora tenemos unos datos limpios y listos para ser usados en nuestro modelo. Aunque a nosotros tomamos poco tiempo para la revisión, deberíamos disponer de más tiempo para revisar los datos y realizar una adecuada limpieza. Durante este ejercicio hemos tomado varias decisiones que deben ser mejor evaluadas.

Datos de ausencia y de fondo

Los algoritmos para realizar modelos de distribución de especies, como Bioclim y Domain, usan datos de "presencia" para realizar el modelado y estos no necesitan información de ausencias. Estos modelos son conocidos como modelos de envuelta ambiental, porque justamente se basan en las características ambientales que tiene cada uno de los datos de presencia. Aunque estos modelos fueron bastante utilizados, hoy en día han perdido importancia. Actualmente los métodos de modelado de distribución de especies usan datos de "*ausencia*" o datos de "*fondo*".

Los datos de ausencia se refiere a los sitios donde aunque hemos buscado la especie, efectivamente no la hemos encontrado. Si solo tenemos datos de presencia, podemos usar un método que necesite datos de ausencia, sustituyendo los datos de ausencia con datos de fondo o pseudo-ausencias. En este caso denominamos pseudo-ausencias o datos de fondo porque vamos a decir que en un sitio determinado no existe la especie aunque realmente no hemos realizado un muestreo para determinar que no está. Aunque, los datos de fondo y pseudo-ausencias hacen referencia a lo mismo, la forma de obtener estos datos es diferente. Preferimos el concepto de fondo porque requiere menos suposiciones y tiene algunos métodos estadísticos coherentes para lidiar con la "superposición" entre la presencia y los puntos de fondo (por ejemplo, Ward et al. 2009; Phillips y Elith, 2011).

El paquete **dismo** tiene una función para muestrear puntos aleatorios (datos de fondo) de un área de estudio. Puede usar una "máscara" para excluir el área no terrestre que no tiene datos (NA). También se puede usar una "extensión" para restringir la generación de puntos a un área más pequeña.

A continuación cargaremos unos datos raster (más adelante hablaremos de ello) que usaremos como mascara, básicamente generaremos unos puntos aleatorios donde el raster tenga datos. Usaremos la función *randomPoints* para generar los puntos aleatorios en esta mascara, además generaremos un cuadrante más pequeño donde se generen los datos de fondo, usaremos el argumento "ext" de la misma función.

```
# Seleccionamos 500 puntos aleatorios
#aseguramos tener los mismos datos aleatorios
set.seed(1963)
#Generamos los datos aleatorios
bg <- randomPoints(mask, 500 )

#generamos un cuadrante donde generar los datos aleatorios
e <- extent(-85, -70, -10, 5)
bg2 <- randomPoints(mask, 50, ext=e)

# set up the plotting area for two maps
par(mfrow=c(1,2), mar=c(3,3,1,1))
plot(!is.na(mask), legend=FALSE)
points(bg, cex=0.3, pch=21)
# now we repeat the sampling, but limit
# the area of sampling using a spatial extent
plot(!is.na(mask), legend=FALSE)
plot(e, add=TRUE, col='red')
points(bg2, cex=0.3, pch=21)
```

Hay varios enfoques que uno podría usar para muestrear puntos de "pseudo-ausencia", es decir, puntos de un área más restringida que los datos de "fondo". VanDerWal et al. (2009) proponen realizar un muestreo en

un radio alrededor de los puntos de presencia. Vamos a usar los datos de *Desmodus rotundus* que habíamos limpiado previamente para generar los radios.

Lo primero que haremos es cambiar los datos que se encuentran como un arreglo de datos (data.frame) y los convertiremos en datos espaciales (SpatialPointsDataFrame)

```
# Generamos círculos con un radio de 50 km
# usamos el objeto tipo SpatialPointDataFrame
x <- circles(desRep, d=50000, lonlat=TRUE)
## Lo convertimos en un polígono
pol <- polygons(x)
```

Lo función *polygons* elimina las zonas en las cuales los círculos se superponen. Finalmente, tomamos una muestra aleatoria de puntos dentro de los polígonos que hemos creado. Solo queremos un punto por celda de cuadrícula.

```
# Muestreamos aleatoriamente dentro del polígono
# extraemos 250 puntos
set.seed(34)
samp1 <- spsample(pol, 250, type='random', iter=25)

# Extraemos el valor para cada punto de la mascara
cells <- cellFromXY(mask, samp1)
length(cells) #tenemos 250 puntos los que generamos
```

```
## [1] 250
```

```
#Eliminamos puntos repetidos
#Los puntos que tengan el mismo valor serán aquellos que
#están dentro de una misma celda
cells <- unique(cells)
length(cells) #17 puntos estaban en una misma celda
```

```
## [1] 227
```

```
              #los eliminamos y nos quedan 233 puntos

#obtenemos las coordenadas de esos puntos
xy <- xyFromCell(mask, cells)

##Graficamos
plot(pol, axes=TRUE)
points(xy, cex=0.4, pch=3, col='blue')
```

Algunos puntos aleatorios han quedado fuera del polígono que creamos. Esto pueda pasar ya que al generar coordenadas a partir de las celdas, R nos da el centroide, y esto puede ocasionar que algunos puntos se salgan de la máscara. Podemos seleccionar solo aquellos puntos que se encuentran dentro de la máscara, nuevamente usaremos la función *over*.

```
spxy <- SpatialPoints(na.omit(xy), proj4string=CRS('+proj=longlat +datum=WGS84'))
o <- over(spxy, geometry(x))
xyInside <- xy[!is.na(o), ]
```

Datos ambientales

Datos raster

Los modelos de distribución de especies necesitan, por un lado, los puntos de presencia que acabamos de limpiar y los datos de ausencia que generamos, pero además, necesitamos variables que expliquen esa distribución (variables explicativas). Generalmente las variables explicativas son variables climáticas, geomorfológicas entre otras (clima, suelo, terreno, vegetación, uso del suelo, etc.)

Las variables explicativas generalmente se organizan como archivos de tipo raster (cuadrícula), de tal forma que tendremos tantos raster como variable explicativa a ser usada. Estos datos generalmente se almacenan en archivos en algún tipo de formato SIG. Se pueden utilizar casi todos los formatos relevantes (incluida la cuadrícula ESRI, geoTiff, netCDF, IDRISI). Los archivos ASCII pueden ser usados aunque estos tienden a reducir considerablemente la velocidad de procesamiento. Para cualquier estudio en particular, todas las capas deben tener la misma extensión espacial, resolución, origen y proyección. Si es necesario, use funciones como *crop*, *extend*, *aggregate*, *resample* y *projectRaster* desde el paquete **raster**.

El paquete **dismo** trae cargadas varias variables provenientes de la base de datos de WorldClim (Hijmans et al., 2004) y los datos de biomas terrestres de la WWF (Olsen et al., 2001). A continuación desarrollemos los pasos para extraer los datos del paquete dismo y empaquetarlos en un stack que nos permita procesar en conjunto todas las variables.

```
#Definimos el directorio donde están los archivos
path <- file.path(system.file(package="dismo"), 'ex')
#obtenemos un listado de las variables tipo grd (grid) que están en dismo
#grd$ nos permite obtener todos las variables grid
library(dismo)
files <- list.files(path, pattern='grd$', full.names=TRUE )
files
```

```
## [1] "C:/Users/ciesp/Documents/R/win-library/3.5/dismo/ex/bio1.grd"
## [2] "C:/Users/ciesp/Documents/R/win-library/3.5/dismo/ex/bio12.grd"
## [3] "C:/Users/ciesp/Documents/R/win-library/3.5/dismo/ex/bio16.grd"
## [4] "C:/Users/ciesp/Documents/R/win-library/3.5/dismo/ex/bio17.grd"
## [5] "C:/Users/ciesp/Documents/R/win-library/3.5/dismo/ex/bio5.grd"
## [6] "C:/Users/ciesp/Documents/R/win-library/3.5/dismo/ex/bio6.grd"
## [7] "C:/Users/ciesp/Documents/R/win-library/3.5/dismo/ex/bio7.grd"
## [8] "C:/Users/ciesp/Documents/R/win-library/3.5/dismo/ex/bio8.grd"
## [9] "C:/Users/ciesp/Documents/R/win-library/3.5/dismo/ex/biome.grd"
```

```
#empaquetamos las variables en un stack
predictors <- stack(files)
predictors
```

```
## class      : RasterStack
## dimensions : 192, 186, 35712, 9  (nrow, ncol, ncell, nlayers)
## resolution : 0.5, 0.5  (x, y)
## extent     : -125, -32, -56, 40  (xmin, xmax, ymin, ymax)
```

```
## crs        : +proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0
## names      : bio1, bio12, bio16, bio17, bio5, bio6, bio7, bio8, biome
## min values :  -23,     0,     0,     0,   61, -212,   60,  -66,     1
## max values :  289,  7682,  2458,  1496,  422,  242,  461,  323,    14
```

```
names(predictors)
```

```
## [1] "bio1"  "bio12" "bio16" "bio17" "bio5"  "bio6"  "bio7"  "bio8"  "biome"
```

```
plot(predictors)
```

Vamos a graficar una de las variables y sobreponer los datos geopolíticos e incluir los datos de Desmodus que hemos limpiado.

```
library(maptools)
#generamos un objeto tipo SpatialPoint
desFD <- desmodFD
coordinates(desFD) <- ~lon+lat
projection(desFD) <- CRS('+proj=longlat +datum=WGS84')

# Graficamos la primera variable climática
plot(predictors, 1)
# Adicionamos los límites políticos
#usamos add=TRUE para que se sobreponga sobre el gráfico anterior
plot(wrld_simpl, add=TRUE)
#finalmente adicionamos los datos de Desmodus
points(desFD, col="darkred", pch=3, cex=0.3)
```

Obtener las variables explicativas es un paso muy importante, sobre todo si nuestro enfoque es predictivo y no explicativo. El modelado de distribución geográfica no busca comprender el efecto de las variables explicativas, sino más bien nos interesa tener capacidad para predecir la distribución. De esta forma, es vital tener variables que permitan realizar la predicción, para lo cual es importante conocer la ecología de la especie. Variables que son importantes para una especie pueden ser irrelevantes para otras, ej. una variable de suelo puede ser muy importante para una planta, pero irrelevante para un ave. En el proceso de modelado necesitamos contar con variables relevantes para la especie que estamos modelando.

Extrayendo los datos ambientales

Ahora que tenemos tanto nuestras variables de presencia-ausencia como nuestras variables explicativas, el siguiente paso es extraer los valores de las variables explicativas. Cada punto de ocurrencia y ausencia le corresponde un valor de cada variable explicativa. Usaremos la función *extract* del paquete **raster** para obtener los valores de las variables contenidas en el stack, usaremos tanto los datos de ocurrencia que limpiamos como los datos de fondo que generamos.

```
#extraemos los valores
presvals <- extract(predictors, desFD) #presencias
absvals <- extract(predictors, bg) #datos de fondo
seabsvals <- extract(predictors, xyInside) #pseudo-ausencias

#generamos una variable de presencia ausencia
```

```r
pb <- c(rep(1, nrow(presvals)), rep(0, nrow(absvals)))

#unimos los datos de presencia-ausencia y la juntamos con la variable
#que acabamos de generar
sdmdata <- data.frame(cbind(pb, rbind(presvals, absvals)))
#convertimos en factor la variable biome
sdmdata[,'biome'] = as.factor(sdmdata[,'biome'])
head(sdmdata)
```

```
##   pb bio1 bio12 bio16 bio17 bio5 bio6 bio7 bio8 biome
## 1  1  245  1042   503    77  336  138  199  276     1
## 2  1  202  2500   811   363  261  149  112  198     1
## 3  1  208  1471   654   162  289  121  168  225     1
## 4  1  255  3416  1491   194  330  190  139  248     1
## 5  1  131   846   452    47  229   21  208  135     1
## 6  1  233   679   286    77  356   84  272  276    13
```

Ahora evaluamos las variables que están correlacionadas, es necesario que dentro del modelo evitemos usar variables que muestren colinearidad espacial. Para ello realizamos correlaciones entre variables para determinar que variables están asociadas y poder decidir que variables usar en el modelo.

```r
pairs(sdmdata[,2:5], cex=0.1)
```


Como podemos observar las variables bio12 y bio16 están fuertemente correlacionadas. Por ahora nos quedaremos con las variables que hemos logrado obtener. Por ahora grabamos los datos para que estén disponibles para los siguientes análisis.

```r
saveRDS(sdmdata, "sdm.Rds")
saveRDS(presvals, "pvals.Rds")
```

Ajuste del modelo, predicción y evaluación

Ajuste de un modelo

Ajustar un SDM (modelo espacial) es técnicamente similar a otras técnicas de modelado. Lo que haremos es identificar las variables dependientes en independientes. De esta forma la variable dependiente presencias-ausencias en relación a las variables ambientales. Vamos a ajustar un modelo lineal generalizado con los datos que hemos levantado.

```r
#leemos los datos
library(dismo)
sdmdata <- readRDS("sdm.Rds")
presvals <- readRDS("pvals.Rds")

#Usamos unas pocas variables
m1 <- glm(pb ~ bio1 + bio5 + bio12, data=sdmdata)
summary(m1)
```

```
##
## Call:
## glm(formula = pb ~ bio1 + bio5 + bio12, data = sdmdata)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.5542  -0.2808  -0.2012   0.4465   0.9337
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  8.900e-01  1.002e-01   8.884  < 2e-16 ***
## bio1         2.141e-03  5.023e-04   4.263 2.31e-05 ***
## bio5        -3.394e-03  5.066e-04  -6.700 4.44e-11 ***
## bio12       -3.420e-05  2.431e-05  -1.407     0.16
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.1774249)
##
##     Null deviance: 126.87  on 669  degrees of freedom
## Residual deviance: 118.16  on 666  degrees of freedom
## AIC: 748.8
##
## Number of Fisher Scoring iterations: 2
```

Ahora tenemos nuestro modelo, pero como habíamos dicho, no nos interesa saber si estas variables afectan significativamente la presencia de especies, sino poder realizar predicciones a partir de este modelo.

Predicciones

Usaremos el modelo para predecir la probabilidad de ocurrencia de esta especie en unas zonas con diferentes condiciones ambientales. Usaremos la función *predict* para proyectar la probabilidad de ocurrencia.

```
#generamos unos valores para cada una de las
#variables usadas en el modelo

bio1 = c(40, 150, 200)
bio5 = c(60, 115, 290)
bio12 = c(600, 1600, 1700)

#Las unimos en un data.frame
pd = data.frame(cbind(bio1, bio5, bio12))
pd
```

```
##   bio1 bio5 bio12
## 1   40   60   600
## 2  150  115  1600
## 3  200  290  1700
```

```
#predecimos la probabilidad de ocurrencia en esos sitios
predict(m1, pd)
```

```
##         1         2         3
## 0.7514777 0.7661183 0.2757831
```

Como vemos la probabilidad de ocurrencia en los sitios 1 y 2 son mayores al sitio 3. Pero, ¿Cómo las variables influencian la ocurrencia? Podemos utilizar la función *response* para ver cómo cada una de estas variables determina la ocurrencia.

```
response(m1)
```

Como vemos bio5 tiene un efecto más fuerte sobre la probabilidad de ocurrencia de la especie.

Ahora, como lo que nos interesa que las probabilidades de ocurrencia se muestren en el espacio, utilizaremos el paquete **raster** y la función predict para espacializar el modelo.

```
# Extraemos las variables
predictors <- stack(list.files(file.path(system.file(package="dismo"), 'ex'), pattern='grd$', full.name
names(predictors)
```

```
## [1] "bio1"  "bio12" "bio16" "bio17" "bio5"  "bio6"  "bio7"  "bio8"  "biome"
```

```
#realizamos la predicción en base al modelo 1
p <- predict(predictors, m1)
#graficamos
plot(p)
```

Evaluación del modelo

Como vemos resulta ser bastante sencillo, hacer un modelo y mostrarlo en el espacio. Sin embargo, lo que nos interesa es saber si este modelo predice efectivamente la ocurrencia de las especies. Existen muchas formas de evaluar el ajuste de un modelo y estas son dependientes del modelo que se esté usando. Uno de los métodos más utilizados para realizar una evaluación de los SDM es la validación cruzada (cross-validation).

La validación cruzada consiste en crear un modelo con un conjunto de datos de "entrenamiento" y probarlo con otro conjunto de datos de ocurrencias conocidas, datos de "prueba". Por lo general, los datos de entrenamiento y prueba se crean muestreando aleatoriamente (sin reemplazo) el conjunto de datos que hemos logrado recabar.

Se pueden usar diferentes medidas para evaluar la calidad de una predicción (Fielding y Bell, 1997, Liu et al., 2011; y Potts, para datos de presencia y Elith 2006 para datos de abundancia), dependiendo del objetivo del estudio. Muchas medidas para evaluar modelos basadas en datos de presencia-ausencia son "dependientes del umbral". Eso significa que primero se debe establecer un umbral sobre el cual se considera que la especie ocurre (por ejemplo, 0.5, aunque 0.5 rara vez es una opción sensata, por ejemplo, ver Lui et al. 2005). Los valores pronosticados por encima de ese umbral indican una predicción de "presencia", y los valores por debajo del umbral indican "ausencia".

Las estadísticas más utilizadas, que son independientes del umbral son el coeficiente de correlación y el área bajo la curva del operador receptor (AUROC, generalmente abreviado como AUC). AUC es una medida de correlación de rango. En datos imparciales, un AUC alto indica que los sitios con valores de aptitud predichos altos tienden a ser áreas de presencia conocida y las ubicaciones con valores de predicción modelo más bajos tienden a ser áreas donde no se sabe que la especie esté presente (ausente o un punto aleatorio).

Un puntaje de AUC de 0.5 significa que el modelo es tan bueno como una suposición aleatoria. Ver Phillips et al. (2006) para una discusión sobre el uso de AUC en el contexto de datos de solo presencia en lugar de datos de presencia / ausencia. Normalmente, modelos por arriba de 0.75 de AUC son modelos considerados aceptables.

Para ejemplificar, vamos a realizar un modelo usando Bioclim un modelo que usa solo datos de presencia.

```
#sacamos el 75% de los datos para entrenamiento
samp <- sample(nrow(sdmdata), round(0.75 * nrow(sdmdata)))
traindata <- sdmdata[samp,]

#Eliminamos las ausencias
traindata <- traindata[traindata[,1] == 1, 2:9]

#El 25% restante queda como datos de prueba
testdata <- sdmdata[-samp,]

#Hacemos el modelo con datos de entrenamiento
bc <- bioclim(traindata)

#evaluamos presencias y ausencias
e <- evaluate(testdata[testdata==1,], testdata[testdata==0,], bc)
e
```

```
## class          : ModelEvaluation
## n presences    : 41
## n absences     : 127
## AUC            : 0.6497983
## cor            : 0.1867107
## max TPR+TNR at : 0.04641163
```

```
#Graficamos el resultado
plot(e, 'ROC')
```

Lo que podemos ver es que nuestro modelo no es diferente a una distribución aleatoria, en otras palabras aunque tenemos un modelo este no predice la distribución de esta especie, la predicción es cercana a un modelo de ocurrencia aleatoria. Más adelante volveremos a la evaluación de los modelos.

Métodos de modelamiento

Tipos de algoritmos usados

Una gran cantidad de algoritmos han sido utilizados para desarrollar SDM. Estos algoritmos pueden ser clasificados en tres tipos; i) envuelta, ii) regresión y iii) aprendizaje automático (Machine learning). Los algoritmos de envuelta consideran solamente los datos de presencia. Los otros dos tipos de algoritmos usan datos de presencia y ausencia.

Vamos a utilizar datos de una especie del *Desmodus rotundus* que hemos limpiado. Además, extraeremos los datos de las variables independientes que usaremos y, finalmente, dividiremos los datos en entrenamiento y validación.

```r
library(dismo)
library(maptools)
data(wrld_simpl)

#extraemos los datos bioclimáticos
predictors <- stack(list.files(file.path(system.file(package="dismo"), 'ex'), pattern='grd$', full.names

#obtenemos los datos de cada predictor para nuestros datos de presencia
presvals <- extract(predictors, desFD)
set.seed(0)

#Generamos datos de fondo
backgr <- randomPoints(predictors, 500)

#Extraemos los datos de cada predictor de los datos de fondo
absvals <- extract(predictors, backgr)

#Generamos un vector de presencia ausencia
pb <- c(rep(1, nrow(presvals)), rep(0, nrow(absvals)))

#Unimos todo
sdmdata <- data.frame(cbind(pb, rbind(presvals, absvals)))
sdmdata[,'biome'] <- as.factor(sdmdata[,'biome'])
```

Debido a que algunos algoritmos no admiten variables categóricas eliminaremos del stack la variable de bioma.

```r
pred_nf <- dropLayer(predictors, 'biome')
```

Ahora dividimos los datos en entrenamiento y prueba.

```r
set.seed(0)

#generamos 5 grupos aleatorios
group <- kfold(desmodFD, 5)

#Escogemos un grupo de prueba y el resto de entrenamiento
pres_train <- desmodFD[group != 1, ]
pres_test <- desmodFD[group == 1, ]

#Restringimos la predicción
ext <- extent(-110, -32, -33, 30)
```

Generamos los datos de fondo que deberán ser separados entre entrenamiento y prueba. Para evitar que los datos de entrenamiento tengan NA o estén fuera del marco establecido vamos a usar el argumento *extf* que nos permite limitar la generación de los puntos de fondo hasta un 12.5% del limite.

```r
#Generamos los datos de fondo
set.seed(10)
backg <- randomPoints(pred_nf, n=1000, ext=ext, extf = -1.25)
colnames(backg) <- c('lon', 'lat')
group <- kfold(backg, 5)
```

```
#separamos datos de entrenamiento y prueba
backg_train <- backg[group != 1, ]
backg_test <- backg[group == 1, ]
```

Ahora graficamos la información que hemos obtenido para confirmar que todo esté bien. Usaremos uno de los predictores como marco de la figura.

```
#Extraemos los datos del raster
r <- raster(pred_nf, 1)

#Graficamos el raster
plot(!is.na(r), col=c('white', 'light grey'), legend=FALSE)

#graficamos la ventana
plot(ext, add=TRUE, col='red', lwd=2)
#graficamos los puntos de fondo
points(backg_train, pch='-', cex=0.9, col='yellow')
points(backg_test, pch='-',  cex=0.9, col='black')
#graficamos los puntos de ocurrencia
points(pres_train$lon, pres_train$lat, pch= '+', col='green')
points(pres_test$lon, pres_test$lat, pch='+', col='blue')
```

Al parecer todo está donde debería estar, ahora podemos ajustar los diferentes modelos utilizando la información que hemos preparado.

Algoritmos de Envuelta

A continuación vamos a usar tres diferentes algoritmos para realizar nuestros modelos; BIOCLIM, DOMAIN y distancia de MAHALANOBIS. Como lo mencionamos anteriormente estos algoritmos usan únicamente los datos de presencia. Aunque estos modelos han sido ampliamente utilizados, según algunos autores, estos modelos son menos efectivos que otros, particularmente cuando son utilizados para evaluar cambio climático. El algoritmo BIOCLIM calcula la similitud de una ubicación comparando los valores de las variables ambientales en cualquier ubicación con una distribución porcentual de los valores en ubicaciones conocidas de ocurrencia ("sitios de entrenamiento").

```
#Ajustamos el algoritmo a nuestros datos
bc <- bioclim(pred_nf, pres_train[,c("lon","lat")])
#evaluamos el modelo
e <- evaluate(pres_test[,c("lon","lat")], backg_test[,c("lon","lat")], bc, pred_nf)
e
```

```
## class          : ModelEvaluation
## n presences    : 34
## n absences     : 200
## AUC            : 0.6017647
## cor            : 0.04966259
## max TPR+TNR at : 0.05872353
```

Como vemos el modelo generado con BIOCLIM no ajusta adecuadamente (AUC 0.61).

```r
#Definimos a que nivel se realizará el corte de presencia-ausencia
tr <- threshold(e, 'spec_sens')
tr
```

```
## [1] 0.05872353
```

```r
#Extrapolamos los datos a un área de interés
pb <- predict(pred_nf, bc, ext=ext, progress='')

#graficamos el modelo
par(mfrow=c(1,2))
plot(pb, main='Bioclim, raw values')
plot(wrld_simpl, add=TRUE, border='dark grey')
plot(pb > tr, main='presence/absence')
plot(wrld_simpl, add=TRUE, border='dark grey')
points(pres_train, pch='+')
```

El algoritmo de Domain (Carpenter et al. 1993) calcula la distancia de Gower entre las variables ambientales en cualquier ubicación y de estas en las ubicaciones de ocurrencia ("sitios de entrenamiento"). A continuación, ajustamos un modelo de dominio, lo evaluamos y hacemos una predicción. Mapeamos la predicción, así como un mapa clasificado subjetivamente en presencia / ausencia.

```r
#generamos el modelo
dm <- domain(pred_nf, pres_train[,c("lon","lat")])

#lo evaluamos
e <- evaluate(pres_test[,c("lon","lat")], backg_test, dm, pred_nf)
e
```

```
## class          : ModelEvaluation
## n presences    : 34
## n absences     : 200
## AUC            : 0.5425
## cor            : 0.1253585
## max TPR+TNR at : 0.4919958
```

```r
#predecimos en el espacio
pd = predict(pred_nf, dm, ext=ext, progress='')

#graficamos el modelo
par(mfrow=c(1,2))
plot(pd, main='Domain, raw values')
plot(wrld_simpl, add=TRUE, border='dark grey')
tr <- threshold(e, 'spec_sens')
plot(pd > tr, main='presence/absence')
plot(wrld_simpl, add=TRUE, border='dark grey')
points(pres_train, pch='+')
```

Finalmente ajustamos el algoritmo Malahanobis. La función *mahal* implementa un modelo de distribución de especies basado en la distancia de Mahalanobis (Mahalanobis, 1936). La distancia de Mahalanobis tiene en cuenta las correlaciones de las variables en el conjunto de datos, y no depende de la escala de mediciones.

```
#Ajustamos el modelo
mm <- mahal(pred_nf, pres_train[,c("lon","lat")])

#lo evaluamos
em <- evaluate(pres_test[,c("lon","lat")], backg_test, mm, pred_nf)
em
```

```
## class          : ModelEvaluation
## n presences    : 34
## n absences     : 200
## AUC            : 0.7479412
## cor            : 0.1532754
## max TPR+TNR at : -1.133013
```

```
#predecimos
pm = predict(pred_nf, mm, ext=ext, progress='')

#Los graficamos
par(mfrow=c(1,2))
pm[pm < -10] <- -10
plot(pm, main='Mahalanobis distance')
plot(wrld_simpl, add=TRUE, border='dark grey')
tr <- threshold(e, 'spec_sens')
plot(pm > tr, main='presence/absence')
plot(wrld_simpl, add=TRUE, border='dark grey')
points(pres_train, pch='+')
```

Modelos de regresiones

Los modelos de regresiones se ajustan a los datos de presencia y ausencia (fondo). No podemos ajustar el modelo con un RasterStack y puntos. Lo que haremos es extraer los valores de los datos ambientales y ajustar los modelos con estos valores.

```
#pegamos los datos de presencia y ausencia
#Las coordenadas
train <- rbind(pres_train[,c("lon","lat")], backg_train[,c("lon","lat")])
#la variable de ocurrencia
pb_train <- c(rep(1, nrow(pres_train)), rep(0, nrow(backg_train)))

#Extraemos los datos ambientales de entrenamiento
envtrain <- extract(predictors, train)
envtrain <- data.frame( cbind(pa=pb_train, envtrain) )
envtrain[,'biome'] = factor(envtrain[,'biome'], levels=1:14)
head(envtrain)
```

```
##   pa bio1 bio12 bio16 bio17 bio5 bio6 bio7 bio8 biome
```

```
## 1  1  245  1042   503    77  336  138  199  276     1
## 2  1  208  1471   654   162  289  121  168  225     1
## 3  1  255  3416  1491   194  330  190  139  248     1
## 4  1  131   846   452    47  229   21  208  135     1
## 5  1  270  1190   501   141  322  216  106  268    13
## 6  1  210  1429   638    81  302  127  175  218     3
```

```
#Extraemos los datos ambientales de evaluación
testpres <- data.frame( extract(predictors, pres_test[,c("lon","lat")]) )
testbackg <- data.frame( extract(predictors, backg_test[,c("lon","lat")]) )
testpres[ ,'biome'] = factor(testpres[ ,'biome'], levels=1:14)
testbackg[ ,'biome'] = factor(testbackg[ ,'biome'], levels=1:14)
```

Modelos Lineales Generalizados Un modelo lineal generalizado (GLM) es una generalización de la regresión de mínimos cuadrados ordinarios. Los modelos se ajustan utilizando la máxima verosimilitud y permitiendo que el modelo lineal se relacione con la variable de respuesta a través de una función de enlace y permitiendo que la magnitud de la varianza de cada medición sea una función de su valor predicho. Dependiendo de cómo se especifique un GLM, puede ser equivalente a la regresión lineal (múltiple), la regresión logística o la regresión de Poisson. Ver Guisan et al (2002) para una visión general del uso de GLM en el modelado de distribución de especies.

A continuación ajustaremos dos modelos lineales generalizados con diferente función de enlace.

```
#Usamos una regresión logística
gm1 <- glm(pa ~ bio16 + bio5 + bio6 + bio17 + bio12,
           family = binomial(link = "logit"), data=envtrain)
summary(gm1)
```

```
##
## Call:
## glm(formula = pa ~ bio16 + bio5 + bio6 + bio17 + bio12, family = binomial(link = "logit"),
##     data = envtrain)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6339  -0.5719  -0.4226  -0.2666   2.6045
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  2.9804205  0.6866848   4.340 1.42e-05 ***
## bio16        0.0050720  0.0014054   3.609 0.000307 ***
## bio5        -0.0190642  0.0026575  -7.174 7.30e-13 ***
## bio6         0.0036771  0.0019862   1.851 0.064118 .
## bio17        0.0008156  0.0018642   0.437 0.661761
## bio12       -0.0019073  0.0007152  -2.667 0.007655 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 775.88  on 935  degrees of freedom
## Residual deviance: 679.67  on 930  degrees of freedom
## AIC: 691.67
##
```

```
## Number of Fisher Scoring iterations: 5
```

```
#Ajustamos un GLM con una función gaussian

gm2 <- glm(pa ~  bio1+bio5 + bio6 + bio7 + bio8 + bio12 + bio16 + bio17,
           family = gaussian(link = "identity"), data=envtrain)

#Evaluamos el modelo 1
ge1 <- evaluate(testpres, testbackg, gm1)
ge1
```

```
## class          : ModelEvaluation
## n presences    : 34
## n absences     : 200
## AUC            : 0.7447059
## cor            : 0.3147866
## max TPR+TNR at : -1.825892
```

```
#Evaluamos el modelo 2
ge2 <- evaluate(testpres, testbackg, gm2)
ge2
```

```
## class          : ModelEvaluation
## n presences    : 34
## n absences     : 200
## AUC            : 0.7266176
## cor            : 0.313569
## max TPR+TNR at : 0.1751886
```

```
#Predecimos la distribución con el primer modelo
pg1 <- predict(predictors, gm1, ext=ext)
pg2 <- predict(predictors, gm2, ext=ext)
#Graficamos
par(mfrow=c(1,2))
plot(pg1, main='GLM/logistic, raw values')
plot(wrld_simpl, add=TRUE, border='dark grey')
tr <- threshold(ge1, 'spec_sens')
plot(pg1 > tr, main='presence/absence')
plot(wrld_simpl, add=TRUE, border='dark grey')
points(pres_train, pch='+')
points(backg_train, pch='-', cex=0.25)
```

Modelos de aprendizaje automático

Dentro de los algoritmos de aprendizaje automatizado MaxEnt (abreviatura de "Entropía máxima"; Phillips et al., 2006) es el algoritmo SDM más utilizado. Maxent se encuentra implementado en Java pero puede ser ejecutado desde el paquete **dismo**. Si se encuentra con una versión de R superior a 3.5, Maxent no se encuentra disponible para esta versión. Sin embargo, puede descargar Maxent haciendo clic aquí. Una vez descargada esta versión vaya a la libreria de R, busque el paquete dismo y la carpeta java, copie y peque el ejecutable de maxent en esta carpeta. La dirección debería ser similar a esta "….R/win-library/3.5/dismo/java"

```r
#Cargamos maxent
maxent()
```

```
## Loading required namespace: rJava

## This is MaxEnt version 3.4.1
```

```r
#Ajustamos el modelo
#Es necesario informar a maxent las variables que se encuentran como factores
mx <- maxent(predictors, pres_train[,c("lon", "lat")], factors='biome')

#Graficamos las variables y su importancia
plot(mx)
```

```r
response(mx)
```

```r
#Realizamos la evaluación del modelo
emx <- evaluate(pres_test[,c("lon", "lat")], backg_test[,c("lon", "lat")], mx, predictors)
emx
```

```
## class          : ModelEvaluation
## n presences    : 34
## n absences     : 200
## AUC            : 0.8197059
## cor            : 0.5213534
## max TPR+TNR at : 0.6323841
```

```r
#Predecimos
px <- predict(predictors, mx, ext=ext, progress='')

#Graficamos
par(mfrow=c(1,2))
plot(px, main='Maxent, raw values')
plot(wrld_simpl, add=TRUE, border='dark grey')
tr <- threshold(e, 'spec_sens')
plot(px > tr, main='presence/absence')
plot(wrld_simpl, add=TRUE, border='dark grey')
points(pres_train, pch='+')
```

Otro método que durante los últimos años se ha desarrollado es el método Random Forest (Breiman, 2001b), este es una extensión de los árboles de clasificación y regresión (CART; Breiman et al., 1984). En R se implementa en la función 'randomForest' en un paquete con el mismo nombre. La función randomForest puede tomar una fórmula o, en dos argumentos separados, un marco de datos con las variables predictoras y un vector con la respuesta. Si la variable de respuesta es un factor (categórico), randomForest hará una clasificación, de lo contrario hará una regresión. Mientras que con el modelo de distribución de especies a menudo estamos interesados en la clasificación (la especie está presente o no), el uso de la regresión proporciona mejores resultados. rf1 hace regresión, rf2 y rf3 hacen clasificación (son exactamente los mismos modelos). Consulte la función tuneRF para optimizar el procedimiento de ajuste del modelo.

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.5.3
```

```
#Ajustamos el modelo
model <- pa ~ bio16 + bio5 + bio6 + bio17 + bio12
rf1 <- randomForest(model, data=envtrain)
```

```
## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values. Are you sure you want to do regression?
```

```
model <- factor(pa) ~ bio16 + bio5 + bio6 + bio17 + bio12
rf2 <- randomForest(model, data=envtrain)
rf3 <- randomForest(envtrain[,1:8], factor(pb_train))

#Evaluamos el modelo usamos unicamente el primero
erf <- evaluate(testpres, testbackg, rf1)
erf
```

```
## class          : ModelEvaluation
## n presences    : 34
## n absences     : 200
## AUC            : 0.7736765
## cor            : 0.3843336
## max TPR+TNR at : 0.1684238
```

```
#predecimos
pr <- predict(predictors, rf1, ext=ext)

#graficamos
par(mfrow=c(1,2))
plot(pr, main='Random Forest, regression')
plot(wrld_simpl, add=TRUE, border='dark grey')
tr <- threshold(erf, 'spec_sens')
plot(pr > tr, main='presence/absence')
plot(wrld_simpl, add=TRUE, border='dark grey')
points(pres_train, pch='+')
points(backg_train, pch='-', cex=0.25)
```

Support Vector Machines (SVMs; Vapnik, 1998) aplica un método lineal simple a los datos, pero en un espacio de características de alta dimensión no relacionado linealmente con el espacio de entrada, pero en la práctica, no implica ningún cálculo en esa alta dimensión del espacio. Esta simplicidad combinada con el rendimiento de vanguardia en muchos problemas de aprendizaje (clasificación, regresión y detección de novedad) ha contribuido a la popularidad de la SVM (Karatzoglou et al., 2006). Fueron utilizados por primera vez en modelos de distribución de especies por Guo et al. (2005). Existen varias implementaciones de svm en R. Las implementaciones más útiles en nuestro contexto son probablemente la función “ksvm” en el paquete “kernlab”.

```
library(kernlab)
```

```
## Warning: package 'kernlab' was built under R version 3.5.2
```

```
##
## Attaching package: 'kernlab'
```

```
## The following objects are masked from 'package:raster':
##
##     buffer, rotated
```

```
svm <- ksvm(pa ~ bio16 + bio5 + bio6 + bio17 + bio12, data=envtrain)
esv <- evaluate(testpres, testbackg, svm)
esv
```

```
## class          : ModelEvaluation
## n presences    : 34
## n absences     : 200
## AUC            : 0.7601471
## cor            : 0.3232351
## max TPR+TNR at : 0.03340697
```

```
#Hacemos la predicción
ps <- predict(predictors, svm, ext=ext)

#Graficamos
par(mfrow=c(1,2))
plot(ps, main='Support Vector Machine')
plot(wrld_simpl, add=TRUE, border='dark grey')
tr <- threshold(esv, 'spec_sens')
plot(ps > tr, main='presence/absence')
plot(wrld_simpl, add=TRUE, border='dark grey')
points(pres_train, pch='+')
points(backg_train, pch='-', cex=0.25)
```

Combinando predicciones del modelo

En lugar de confiar en un solo "mejor" modelo, algunos autores (por ejemplo, Thuillier, 2003) han abogado por utilizar muchos modelos y aplicar algún tipo de promedio de modelos. Vea el paquete biomod2 para una implementación. Por supuesto, puede implementar estos enfoques usted mismo. A continuación se muestra un ejemplo muy breve. Primero hacemos un RasterStack de nuestras predicciones de modelos individuales:

```
#modelos con AUC mayor a 0.70
models <- stack( pg1, pg2,px,pr,ps)
names(models) <- c("glmLog","glmGau", "maxent", "rf", "svm")
plot(models)
```

```r
#Calculamos la media
m <- mean(models)
plot(m, main='average score')
```

```r
#como una media pesada por el AUC

auc <- sapply(list(emx,ge1, ge2, erf, esv), function(x) x@auc)
w <- (auc-0.5)^2
m2 <- weighted.mean( models, w)
plot(m2, main='weighted mean of three models')
```