Datos espaciales poblacionales

Paula Vargas Pellicer

05/05/2022

En esta clase, nos centraremos en cómo formatear, manipular y visualizar datos de tendencia de población y ocurrencia de especies. Usaremos los paquetes tidyr y dplyr para limpiar los datos y calcular nuevas variables. Usaremos también ggplot2 para hacer lindos mapas simples de registros de ocurrencia, visualizar algunas tendencias en el tiempo y luego organizaremos todos nuestros gráficos juntos usando el paquete gridExtra.

1. Descargar, formatear y manipular datos de biodiversidad

Trabajaremos con datos de ocurrencia de la ballena beluga del Fondo de Información sobre Biodiversidad Global y datos de población para la misma especie de la base de datos Living Planet, los cuales son conjuntos de datos disponibles públicamente.

```
setwd("tu_ruta_de_archivo")

# Packages ----
library(readr)
library(tidyr)
library(dplyr)

library(broom)
library(ggplot2)
library(ggthemes)
library(mapdata)

library(maps)
library(rgbif)
library(ggrepel)
library(png)
library(gridExtra)
```

Haz tu propio tema ggplot2

```
# Personaliza el tema, cambia los números y colores a tu gusto
theme_marine <- function(){
  theme_bw() +
    theme(axis.text = element_text(size = 16),
          axis.title = element_text(size = 20),
          axis.line.x = element_line(color="black"),
          axis.line.y = element_line(color="black"),
          panel.border = element_blank(),
          panel.grid.major.x = element_blank(),
          panel.grid.minor.x = element_blank(),
          panel.grid.minor.y = element_blank(),
          panel.grid.major.y = element_blank(),
          plot.margin = unit(c(1, 1, 1, 1), units = , "cm"),
          plot.title = element_text(size = 20),
          legend.text = element_text(size = 12),
          legend.title = element_blank(),
          legend.position = c(0.9, 0.9),
          legend.key = element_blank(),
          legend.background = element_rect(color = "black",
                                           fill = "transparent",
                                           size = 2, linetype="blank"))
}
```

Cargar datos de ocurrencia de especies y tendencias de población

Los datos están en formato .RData, éstos son más rápidos de usar ya que están más comprimidos. El inconveniente es que solo se pueden usar dentro de R, mientras que los archivos .csv son más transferibles.

```
load("beluga.RData")
belugas<- as.data.frame(beluga)

# Carga los datos de cambios poblacionales para las especies marinas de la ba
se de datos Living Planet
load("marine.RData")
```

Formateo de datos

El objeto beluga contiene cientos de columnas con información sobre los registros de GBIF. Para acelerar nuestro análisis, podemos seleccionar solo las que necesitamos usando la función de selección del paquete dplyr, que selecciona solo las columnas que solicitamos.

```
# Simplifiquemos los datos de ocurrencia
belugas <- belugas %>% dplyr::select(key, name, decimalLongitude, decimalLati
tude, year, individualCount, country)
```

Especificamos que queremos la función de selección exactamente del paquete dplyr y no de ningún otro paquete que hayamos cargado usando dplyr::select. De lo contrario, es posible que obtengas el siguiente error: #Error in (function (classes, fdef, mtable) : unable to find an inherited method for function select for signature "grouped_df"

Dar formato y manipular el conjunto de datos de cambio de población

A continuación, seguiremos algunos pasos consecutivos para formatear los datos de cambio de población, excluir los valores de NA y preparar los datos de abundancia para el análisis. Para este propósito, vamos a usar las tuberías.__Pipes (%>%)

Los datos de cambio de población están en un formato ancho: cada fila contiene una población que ha sido monitoreada a lo largo del tiempo y hacia la derecha de la base de datos, hay muchas columnas con estimaciones de población para cada año. Para hacer que estos datos estén "ordenados" (una columna por variable) podemos usar el método de gather() para transformar los datos de modo que haya una nueva columna que contenga todos los años para cada población y una columna adyacente que contenga todas las estimaciones de población para esos años.

Tomaremos nuestro conjunto de datos marino original, filtraremos para incluir solo las poblaciones de beluga y crearemos una nueva columna llamada año, la llenaremos con los nombres de las columnas de los números 26 a 70 (26:70) y luego usaremos los datos de estas columnas para hacer otra columna llamada abundancia.

También deberíamos escalar los datos de población, porque, dado que los datos provienen de muchas especies, las unidades y la magnitud de los datos son muy diferentes. Imagina peces diminutos cuya abundancia es de millones y grandes carnívoros cuya abundancia es mucho menor. Al escalar los datos, también los estamos normalizando para que más adelante podamos usar modelos lineales con una distribución normal para cuantificar el cambio general experimentado en la población.

```
# veamos los datos
head(marine)

# Formatea los datos de cambios poblacionales

beluga.pop <- marine %>% filter(Genus == "Delphinapterus") %>%  # Selecciona
solo poblaciónes de beluga
  gather(key = "year", value = "abundance", 26:70) %>%  # cambiar a formato
largo
  filter(is.na(abundance) == FALSE) %>%  # quiatr filas vacías
  group_by(id) %>%   # Agrupa filas para que cada grupo sea una población
mutate(scalepop = (abundance-min(abundance))/(max(abundance)-min(abundance)))
```

```
%>%  # escala la abundancia de 0 a 1
  filter(length(unique(year)) > 4) %>% # Solo incluye poblaciones monitoreada
s al menos 5 veces
  ungroup()  # Elimina el agrupamiento
```

Debido a que los nombres de las columnas están codificados como caracteres, cuando convertimos los nombres de las columnas (1970, 1971, 1972, etc.) en filas, R colocó automáticamente una X delante de los números para obligarlos a permanecer como caracteres. No queremos eso, así que, para convertir el año en una variable numérica, puede usar la función parse_number del paquete readr.

```
# Explora la base de datos
str(beluga.pop)


# Elimina la X
beluga.pop$year <- parse_number(beluga.pop$year)
```

Cuantificar el cambio de población

Ajustaremos modelos lineales simples (abundancia a lo largo del tiempo para cada población, abundancia ~ año) para obtener una medida del cambio general experimentado por cada población durante el período en que fue monitoreada. Extraeremos la pendiente, es decir, la estimación del término de año para cada población. También podemos hacer esto en una tubería, lo que hace que el análisis sea eficiente. Aquí estamos analizando cinco poblaciones, pero también puedes hacerlo por miles.

```
# Calcula los cambios poblacionales con un modelo lineal

beluga.slopes <- beluga.pop %>%
  group_by(Location.of.population, Decimal.Latitude, Decimal.Longitude, id)
%>%
  do(mod = lm(scalepop ~ year, data = .))

# extrae los coeficientes usando tidy() del paquete broom
test <- beluga.pop %>%
  group_by(Location.of.population, Decimal.Latitude, Decimal.Longitude, id)
%>%
  data.frame(., as.list(coef(lm(scalepop ~ year, data = .)))) %>%
  rename_at(30:31, ~c("estimate", "term"))

head(beluga.slopes)

# selecciona las columnas que necesitamos
beluga.slopes <- test %>%
  dplyr::select(Location.of.population, Decimal.Latitude,
                Decimal.Longitude, id, term, estimate)
```

2. Visualización de datos de presencia de especies

```
(beluga.map <- ggplot(beluga, aes(x = decimalLongitude, y = decimalLatitude))
+
   borders("worldHires", ylim = c(40, 100), colour = "gray40", fill = "gray40
", size = 0.3) +
     # selecciona el mapa
   theme_map() +
   geom_point(alpha = 0.5, size = 2, colour = "aquamarine3"))  # alpha contro
la la transparencia 1=totalemente transparente
```

¿Ves alguna ballena beluga donde no debería haber ninguna? Puedes verificar qué dice la función CleanCoordinates como registros potencialmente incorrectos. La función realiza una serie de pruebas, p.ejemplo si hay ceros entre los valores de longitud y latitud, y luego devuelve datos de resumen de si cada registro de ocurrencia falló o pasó cada prueba (es decir, FALSO o VERDADERO).

```
# ** Aquí puedes limpiar los datos de ocurrencia ----

# puedes checar las coordenadas para todas las ocurrencias
# carga un objeto con una línea de costa que tiene un buffer para ver si los
puntos están en tierra o mar


load("buffland_1deg.rda")


#Por ejemplo, si quisieramos conserver records de avistamientos que fueron to
mados desde la costa

# Usa clean_coordinates() del paquete CoordinateCleaner


beluga.coord.test <- clean_coordinates(beluga, lon = "decimalLongitude", lat
= "decimalLatitude", species = "name", tests = c("outliers", "seas", "zeros")
, outliers_method = "distance", outliers_td = 5000,seas_ref = buffland)


species = "" se refiere al nombre la la columna que tiene el nombre de las es
pecies

# Por default, los puntos extremos son las ocurrencias que están más lejos de
1000km que cualquier otra ocurrencia
# Puedes cambiar eso usando outliers.td() con el valor que quieras

#Probemos si las ocurrencias están en tierra o en mar y si hay ceros en lat o
long
```

```
head(beluga.coord.test)

# Algunas ocurrencias están en la tierra
# Extrae solo las ocurrencias clasificadas como TRUE usando value = "clean"


beluga.clean <- clean_coordinates(beluga, lon = "decimalLongitude", lat = "de
cimalLatitude",species = "name", tests = c("outliers", "seas", "zeros"),outli
ers_method = "distance", outliers_td = 5000,seas_ref = buffland, value = "cle
an")
```

Ahora podemos actualizar nuestro mapa intercambiando el objeto beluga con el objeto beluga.clean para ver qué diferencia hizo limpiar los datos de coordenadas. Al limpiar datos de coordenadas para tus propios análisis, puedes incluir más pruebas, p.ejemplo si estás realizando un análisis solo en México, puedes verificar si los puntos están realmente en el México. Como es habitual en la investigación, la decisión sobre qué incluir o excluir es totalmente tuya y la función clean_coordinates ofrece una forma de informar tu decisión.


Haz un mapa las ocurrencias limpias (si lograste bajar el paquete)

Curiosamente, todavía aparecen algunos puntos de tierra. Las pruebas eliminaron los valores atípicos obvios, como el registro de beluga cerca de África. Podemos excluir manualmente puntos que creemos que no son reales si lo deseamos, p.ejemplo el punto terrestre de Groenlandia es bastante distinto y fácil de eliminar.

```
beluga<- as.data.frame(beluga)
greenland <- filter(beluga, country == "Greenland")

# Selecciona una coordenada única para el las ocurrencias del mismo lugar

greenland <- dplyr::select(greenland, decimalLongitude, decimalLatitude) %>%
  distinct()

head(greenland)

# El punto a excluir es uno que está muy al este

# Longitude -46.00000  Latitude 65.00000

# encuentra qué filas se tienen ese valor usando which()
which(beluga$decimalLongitude == -46 & beluga$decimalLatitude == 65)

beluga.base <- beluga[-c(1421, 1422, 1423, 1424, 1530),]


# O puedes filtrar esos puntos de esta manera:
```

```
 beluga.pipe <- beluga %>% filter(decimalLongitude != -46 | decimalLatitude !
= 65)
```

Podríamos continuar con la limpieza de ocurrencias hasta que estés certer@ que los datos son correctos, sobre todo si conoces bien a las poblaciones.

Por ahora, pasaremos a más visualización de datos. Personalizaremos nuestro mapa de presencia de belugas, visualizaremos cuándo se recopilaron los registros y cómo algunas de las poblaciones de belugas han cambiado a lo largo del tiempo.

```
# Haz un mapa nuevo que incluya las locaciones de las poblaciónes

(beluga.map.LPI <- ggplot(beluga.base, aes(x = decimalLongitude, y = decimalL
atitude)) +
    borders("worldHires", ylim = c(40, 100), colour = "gray40", fill = "gray4
0", size = 0.3) +
    theme_map() +
    geom_point(alpha = 0.3, size = 2, colour = "aquamarine3") +
    geom_point(data = beluga.slopes, aes(x = Decimal.Longitude, y = Decimal.L
atitude),  # Agrega los puntos de los datos de los cambios de poblaciones
size = 4, colour = "tan1"))

# Aquí especificaste de dónde vienen los datos
```

Vamos a a gregar etiquetas para los sitios de monitoreo de la beluga. Primero hay que revisar que los nombres sean consistentes en la base de datos

```
# checa los nombres de los sitios
print(beluga.slopes$Location.of.population)

# Haz los nombres consistentes


beluga.slopes$Location.of.population <- recode(beluga.slopes$Location.of.popu
lation,
                                               "Cook Inlet stock, Alaska" = "
Cook Inlet stock")
beluga.slopes$Location.of.population <- recode(beluga.slopes$Location.of.popu
lation,
                                               "Eastern Hudson Bay, Quí©bec"
= "Eastern Hudson Bay")
beluga.slopes$Location.of.population <- recode(beluga.slopes$Location.of.popu
lation,
                                               "St. Lawrence estuary populati
on" = "St. Lawrence Estuary")
beluga.slopes$Location.of.population <- recode(beluga.slopes$Location.of.popu
lation,
                                               "St. Lawrence Estuary populati
on" = "St. Lawrence Estuary")
```

```
beluga.slopes$Location.of.population <- recode(beluga.slopes$Location.of.popu
lation,
                                               "St. Lawrence estuary, Canada"
= "St. Lawrence Estuary")

# Checalos
print(beluga.slopes$Location.of.population)
```

Podemos usar la función annotation_custom de ggplot2 para agregar imágenes a nuestros gráficos, por ejemplo, un icono de beluga.

```
# Carga los paquetes para agregar imágenes
packs <- c("png","grid")
lapply(packs, require, character.only = TRUE)

# Carga el ícono de la beluga
icon <- readPNG("beluga_icon.png")

icon <- rasterGrob(icon, interpolate=TRUE)
# Hay que "raterizarlo" para poder graficarlo
```

¡Ahora viene lo que parece un trozo gigantesco de código!

Para agregar las etiquetas: estamos especificando la base de datos para las etiquetas (un sitio tiene tres poblaciones monitoreadas) pero solo queremos la etiqueta una sola vez por lo que seleccionamos data = beluga.slopes[1:3,] para tener las primeras tres filas #y todas las columnas. Estamos especificando el tamaño de las etiquetas y moviendo los puntos en ambos ejes para que no se escondan.

```
(beluga.map.final <- ggplot(beluga.base, aes(x = decimalLongitude, y =
decimalLatitude)) +
    borders("worldHires", ylim = c(40, 100), colour = "gray40", fill =
"gray40", size = 0.3) +
    theme_map() +
    geom_point(alpha = 0.3, size = 2, colour = "aquamarine3") +
    geom_label_repel(data = beluga.slopes[1:3,], aes(x = Decimal.Longitude, y
= Decimal.Latitude,
                                                     label =
Location.of.population),
                     box.padding = 1, size = 5, nudge_x = 1,
                     nudge_y = ifelse(beluga.slopes[1:3,]$id == 13273, 4, -
4),
                     min.segment.length = 0, inherit.aes = FALSE) +
    geom_point(data = beluga.slopes, aes(x = Decimal.Longitude, y =
```

```
Decimal.Latitude + 0.6),
               size = 4, colour = "tan1") +
    geom_point(data = beluga.slopes, aes(x = Decimal.Longitude, y =
Decimal.Latitude - 0.3),
               size = 3, fill = "tan1", colour = "tan1", shape = 25) +
        # Agrega puntos para los cambios de poblaciones
    annotation_custom(icon, xmin = -210, xmax = -120, ymin = 15, ymax = 35) +
# Agrega el icono
    labs(title = "a. Ocurrencias de Beluga GBIF") +  # titulo
    theme(plot.title = element_text(size = 20)))  # tamaño del titulo
```

A continuación, probaremos otros dos tipos de gráficos, un gráfico lineal y algunos diagramas de dispersión con ajustes de modelos lineales. Podemos usar nuestro tema ggplot2 personalizado, theme_marine, para que todos nuestros gráficos tengan un formato consistente y no necesitemos repetir el mismo código,.

```
# Numero de registros de ocurrencias a traves del tiempo
yearly.obs <- beluga %>% group_by(year) %>% tally() %>% ungroup() %>% filter(
is.na(year) == FALSE)

(occurrences <- ggplot(yearly.obs, aes(x = year, y = n)) +
    geom_line(colour = "aquamarine3", size = 1) +
    geom_area(aes(y = n), fill = "aquamarine3") +

    labs(x = NULL, y = "Número de occurrencias\n") +
    theme_marine())
```

Para nuestro conjunto final de gráficos, trazaremos la abundancia de belugas a lo largo del tiempo y un ajuste de modelo lineal del cambio de población para cada población de belugas, parte de la base de datos Living Planet.

```
# Crea un objeto para la población de Hudson Bay
beluga1 <- filter(beluga.pop, id == "13273")

# Elige la pendiente de los cambios de la población par ponerlo en la grafica
, si quieres

print(beluga.slopes$year[beluga.slopes$id == "13273"])

(hudson.bay <- ggplot(beluga1, aes(x = year, y = abundance)) +
    geom_point(shape = 21, fill = "aquamarine3", size = 4) +
            geom_smooth(method = "lm", colour = "aquamarine3", fill = "aquama
rine3", alpha = 0.4) +
            labs(x = "", y = "IndividuOS\n", title = "c. Eastern Hudson Bay\n
") +    theme_marine())
```

Ahora, podemos modificar un poco el código para hacer un gráfico para la población de stock de Cook Inlet.

```
beluga2 <- filter(beluga.pop, id == "2191")

(cook.inlet <- ggplot(beluga2, aes(x = year, y = abundance)) +
    geom_point(shape = 21, fill = "aquamarine3", size = 4) +
    geom_smooth(method = "lm", colour = "aquamarine3", fill = "aquamarine3",
alpha = 0.4) +
    labs(x = "", y = "", title = "d. Cook Inlet stock\n") +
    theme_marine())
```

El último sitio, el estuario de St. Lawrence, ha sido monitoreado por tres estudios diferentes en diferentes períodos de tiempo.

```
# Crea un objeto que tenga las tres poblaciónes del estuario de St. Lawrence
estuary
# usando el operador "|"
beluga3 <- filter(beluga.pop, id == "1950" | id == "4557" | id == "4558")

(st.lawrence.est <- ggplot(beluga3, aes(x = year, y = abundance, shape = as.f
actor(id))) +
    geom_point(fill = "aquamarine3", size = 4) +
    scale_shape_manual(values = c(21, 23, 24)) +
    geom_smooth(method = "lm", colour = "aquamarine3", fill = "aquamarine3",
alpha = 0.4) +
    labs(x = "", y = "", title = "e. Estuario St. Lawrence\n") +
    theme_marine() +
    guides(shape = FALSE))
```

Organiza todos los gráficos en un panel con el paquete gridExtra

La función grid.arrange del paquete gridExtra crea paneles de diferentes gráficos.

```
# Crea el panel con todas las graficas
row1 <- grid.arrange(beluga.map.final, occurrences, ncol = 2, widths = c(1.96
, 1.04))

# Especifica como quieres que sean las proporciones
row2 <- grid.arrange(hudson.bay, cook.inlet, st.lawrence.est, ncol = 3, width
s = c(1.1, 1, 1))

# Cambia los Espacios de las graficas

beluga.panel <- grid.arrange(row1, row2, nrow = 2, heights = c(0.9, 1.1))
```