Jorge López Jiménez

Minería de datos espaciales con R

Introducción

Para esta práctica vamos a necesitar el interpretador de R y para un más fácil manejo de los datos RStudio, lo cual nos ahorrará mucho tiempo.

En esta práctica necesitaremos las librerías siguientes:

- **ggmap:** Extiende las capacidades de ggpplot2 para graficar mapas.
- **rgdal:** Una interface a la popular librería de proceso de datos espaciales de C/C++ gdal.
- **rgeos:** Una interface a la poderosa librería de procesamiento vectorial geos.
- **maptools:** Provee varias funciones de mapeo.
- **tmap:** Nueva paquetería para crear mapas hermosos.

Como vimos en la práctica anterior para instalar cualquier paquetería usaremos la función **library()**, si la función no entrega ninguna cadena de texto significa que habremos importado las librerías necesarias. En caso de que regrese alguna cadena de error, muy probablemente signifique que no estén instaladas las librerías, para lo cual tendríamos que usar la función **install.packages()** la cual recibirá como argumento la librería que tenemos que instalar antes de importar.

Atajo; hay una manera rápida de instalar e importar todas las librerías descritas con sólo un vector de la siguiente manera:

x <- c("ggmap", "rgdal", "rgeos", "maptools", "dplyr", "tidyr", "tmap"**)**

install.packages(x**)** # esto podría tardar algunos minutos

lapply(x, library, character.only = TRUE) # cargará las librerías necesarias

Datos espaciales en R

Ahora lo que haremos será conocer y explorar el formato que usan las bases de datos espaciales para extraer, transformar, cargar (ETL) y graficar los datos.

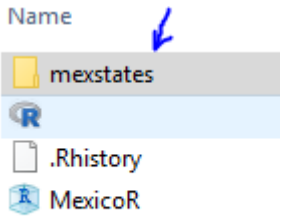
Lo primero es conseguir estas bases de datos. En la información espacial internet tiene muchos recursos gratuitos, libres y bajo licencias de software libre. Uno de los formatos más populares para bases de datos espaciales es 'shapefiles' (realmente es un formato de archivo geométrico).

Dentro del interpretador de R hay muchas maneras de importar este tipo de archivos pero la manera más popular es con la función **readOGR** de la librería **rgdal.**

rgdal es la interfaz de R a la "Geospatial Abstraction Library (GDAL)" la cual es usada por varios software open source del tipo GIS como QGIS, la cual ayuda a R a manejar un gran rango de formatos de archivos.

Descargue un archivo shapefile como el del siguiente enlace:
http://www.arcgis.com/home/item.html?id=ac9041c51b5c49c683fbfec61dc03ba8

Descomprímalo en la carpeta de un nuevo projecto de RStudio de la siguiente manera:



Revise el formato de los archivos que tenga la base de datos espacial y verifique que tenga compatibilidad con readOGR.

Para importar el los datos escribiéremos la función de la siguiente manera:

lnd <- readOGR(dsn = "mexstates", layer = "mexstates")

El primer argumento de la función dsn significa 'data source name' aquí pondremos la ruta de donde encontraremos el archivo espacial, luego layer el cual le pasaremos el nombre del archivo de la base de datos espacial. Note como no tenemos que poner el formato del archivo, esta función inmediatamente identificará la mejor manera de importar estos archivos.

En R no necesitamos definir el nombre del argumento antes de pasárselo dado que el orden en las funciones de los argumentos importa pero es buena práctica definir estos nombres para la legibilidad de las funciones.

La estructura de datos espaciales en R

Una vez creado el objeto espacial lnd podremos ver que está conformado de diferentes ranuras (slots) siendo los que más nos importan @data y @polygons . La ranura @data podríamos verla como una tabla normal en una base de datos, con filas y columnas. Y la ranura @geometry es donde se almacenan los vértices de los polígonos.

Veamos cómo está conformada la ranura de @data

head(lnd@data, n = 2)

Con el argumento n nos traeremos las dos primeras filas de la ranura @data, podremos observar cómo está conformados los datos.

Gráfica básica

Ahora que sabemos cómo está construida la estructura de datos clásica de la ranura de datos, sigue graficar nuestro mapa, lo haremos con uno de las funciones más poderosas y versátiles de R

plot(lnd)

Note como simplemente pasamos como argumento el objeto espacial y automáticamente la función plot sabrá graficarlo y nos entregará una serie de polígonos vectorizados como estos;

Como se puede apreciar en la imagen plot es una función inteligente que te mostrará lo que quieres graficar y bastante personalizable.

Selecciones básicas y espaciales

Ahora el siguiente paso es hacer una selección a los datos con un criterio fijo.

Para la base de datos que importamos vamos a seleccionar los estados que tengan una población mayor a los 8 millones de habitantes del atributo de población:

lnd@data[lnd$POP_ADMIN > 8000000,]

En esa línea de código lo que estamos haciendo es hacer una consulta a la ranura del objeto espacial usando el operador de selección [], el cual recibe dos cosas, el criterio de selección y después de la coma los atributos (columnas que queramos traer) ya sea por su posición en la ranura o por su nombre.

R al ser un lenguaje funcional nos permite guardar sentencias dentro de variables para luego ejecutarlas dentro de subsets, por ejemplo.

Sel <- lnd$POP_ADMIN < 8000000 & lnd$POP_ADMIN > 3000000

lnd@data[sel , c(4,8,11)]

```
# ADMIN_NAME POP_ADMIN     SQKM
2  Nuevo Leon   3370912 65173.05
8     Jalisco   5772704 79851.44
10   Veracruz   6774736 71286.36
11 Guanajuato   4332525 30466.18
17     Puebla   4487672 34365.46
18  Michoacan   3859507 59617.63
25     Oaxaca   3329574 92715.81
27    Chiapas   3524501 73771.48
```

Como podemos ver se pueden unir sentencias en su homónimo en SQL AND con el operador &. Están declaradas en la variable sel, la cual pasamos al operador y le pasamos como segundo argumento un vector que contendrá la dimensión de nuestra tabla y las columnas de la ranura de datos que quisiéramos extraer.

Funciones espaciales

Al igual que varias extensiones espaciales rgeos nos provee de bastantes funciones que podemos usar para hacerle minería a los objetos espaciales. Sacaremos el centroide geométrico de México:

gCentroid(lnd)

```
          x        y
1 -102.5329 23.95046
```

Nos entregará unas coordenadas geométricas del centroide, para poder visualizar estas coordenadas es importante siempre meter lo que sea dentro de plot.

Sin embargo si hacemos lo siguiente:

plot(GCentroid(lnd))

Lo que nos dará de salida será un punto solamente, donde se encuentra el centroide en nuestras coordenadas relativas, entonces primero tenemos que dibujar la serie de polígonos que forman a México:

plot(lnd)

Después agregamos el centroide con la función points() la cual nos permite agregar puntos con coordenadas fijas a la gráfica que tengamos previamente desplegada, se hace de la siguiente manera:

points(gCentroid(lnd), col = “blue”)

Para que nos entregue con esas dos líneas de código un mapa así:

En azul se puede apreciar el centroide geométrico de nuestro mapa, lo cual es perfecto ya que podemos agregar puntos de diferentes colores con el parámetro 'col', parámetro que también existe en plot y lines.

Consultas espaciales 'complejas'

Ahora, ¿y sí queremos el centroide ded cada estado? y ¿qué tal aquellos estados cuyo centroide esté a menos de 600 kilómetros del centriode de la ciudad de México? ¡Hagámoslo!

Lo primero es conseguir consultar sólo el centroide de la ciudad de México, tenemos que seleccionar la fila que se asocie al polígono que buscamos.

Existen muchas maneras de hacer eso pero nosotros usaremos una función que se llama 'grep', funciona muy parecida al programa de UNIX, de la siguiente manera:

lnd@data[grep("Distrito", lnd$ADMIN_NAME),]

Como ya vimos antes, lo que estamos haciendo es seleccionando de la ranura de datos todas las filas que cumplan con una sentencia de selección, esta sentencia en este caso es una función, la función grep(), la cual toma como argumento principalmente dos cosas, una cadena de caracteres y un campo de un dataframe.

En este caso grep regresará el número identificador de la fila o filas que cumplan tener en su campo de nombre administrativo el pedazo de caracteres "Distrito".

¡grep() es tan poderoso que incluso recibe regex como argumento de cadena!, pero eso va más allá del alcance de la práctica, por lo mientras nos quedaremos con que traiga la fila de distrito federal.

Guardaremos nuestra sentencia en una variable para facilitarnos su manejo.

df <- lnd[grep("Distrito", lnd$ADMIN_NAME),]

Nótese como guardamos la sentencia, no fue exactamente igual que cuando la visualizamos, en este caso no sólo seleccionamos la ranura de datos (lnd@data) sino seleccionamos todo el objeto espacial para que se mantenga la relación con sus polígonos.

Perfecto, ahora podemos seleccionar la fila que nos interesa su centroide para comparar a las demás. Probemos graficarlo:

plot(lnd)

points(gCentroid(df), col= "yellow")

Nos entregará el mapa de México con el centroide de CDMX resaltado en amarillo.

El siguiente paso es definir la sentencia de selección por la cual seleccionaremos que los estados tengan una distancia que buscamos lo haremos con la función gDistance() la cual acepta como argumento principal dos geometrías y devuelve la distancia entre ellas

nearDF <- gDistance(gCentroid(df), gCentroid(lnd, byid = TRUE), byid = TRUE) < 6

No hay que intimidarnos por la complejidad de los paréntesis que parece nunca terminan, los lenguajes funcionales siempre se usarán muchos paréntesis, lo que estamos pasando a la variable nearDF es una sentencia de selección, que seleccione filas cuya distancia entre el centroide del DF y el centroide de cada estado sea menor a 600.

La bandera byid que seguramente alienaría a cualquiera, lo único que hace es realizar la misma función pero en vez de hacerlo en la figura general lo hace por cada polígono del cual esté compuesta la geometría grande (por cada estado en este caso).

Podemos ver que estados son esos mediante una selección como habíamos hecho anteriormente así:

lnd@data[nearDF, 4]

La cual nos regresaría los estados cuyo centroide está a 600 o menos kilómetros del DF

```
[1] Tamaulipas       Zacatecas        San Luis Potosi  Jalisco
 [5] Aguascalientes   Veracruz         Guanajuato       Queretaro
 [9] Hidalgo          Puebla           Michoacan        Mexico
[13] Tlaxcala         Colima           Distrito Federal Morelos
[17] Guerrero         Oaxaca
```

Sólo nos trajimos la columna 4 la cual corresponde al nombre del estado.

¡Perfecto!, el resultado se ve coherente, ahora nos faltaría graficarlo, primero crearemos un objeto espacial sólo con los estados que cumplan nuestra sentencia de la siguiente manera:

nearStates <- lnd[c(nearDF), 4]

Nótese cómo vamos a convertir la matriz de selección en un vector con la función c() dado que al seleccionar un objeto espacial no podemos pasar una matriz como argumento, tiene que ser un vector lógico.

Necesitamos graficar el país entero, para esto usaremos una función conocida:

plot(lnd)

Luego graficaremos sobre esta los estados que queremos resaltar.

plot(nearStates, col = “blue”, add = TRUE)

Se active la bandera de ‘add’ para que esta gráfica se adhiera a la anterior, sin esta bandera la gráfica entera (el país completo) se borraría.

Así se ven los estados cuyo centroide está a menos de 600 kilómetros del centride de CDMX.