Mapas en 3D en R

Rodrigo Tapia McClung

15 de enero, 2020

Crear mapas en 3D en R

R *no* es un sistema de información geográfica (SIG), pero se puede usar para crear mapas muy bonitos, tanto en 2D como en 3D. En esta práctica vamos a usar datos espaciales en formato `geojson` para para hacer un mapa en 3D.

Los datos para esta práctica los puedes descargar de aquí. Una vez descargados los datos, descomprime el archivo en alguna carpeta en tu computadora.

Ahora nos cambiamos al directorio de trabajo en donde está un `json` de una zona del país (¿la puedes reconocer?):

```
# Cambiar directorio de trabajo:
setwd("C:/Descargas/escuela-de-metodos-2020/practica3")
```

Y nos aseguramos de tener las librerías que vamos a usar:

```
if (!require("pacman")) install.packages("pacman")
# para macOS, primero instalar https://www.xquartz.org/
# y ejecutarlo antes de usar plot_3d()
# o usar options(rgl.printRglwidget = TRUE)
pacman::p_load(sf, ggplot2, ggmap, rayshader, magick)
```

Ahora podemos leer los datos espaciales:

```
densidad <- st_read("densidad_poblacion.json")
```

```
## Reading layer `densidad_poblacion' from data source `/Users/rtm/pCloud Drive/2020/EM Morelia/practica
## Simple feature collection with 341 features and 25 fields
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: 248185.7 ymin: 2172364 xmax: 279045.9 ymax: 2187419
## epsg (SRID):    32614
## proj4string:    +proj=utm +zone=14 +datum=WGS84 +units=m +no_defs
```

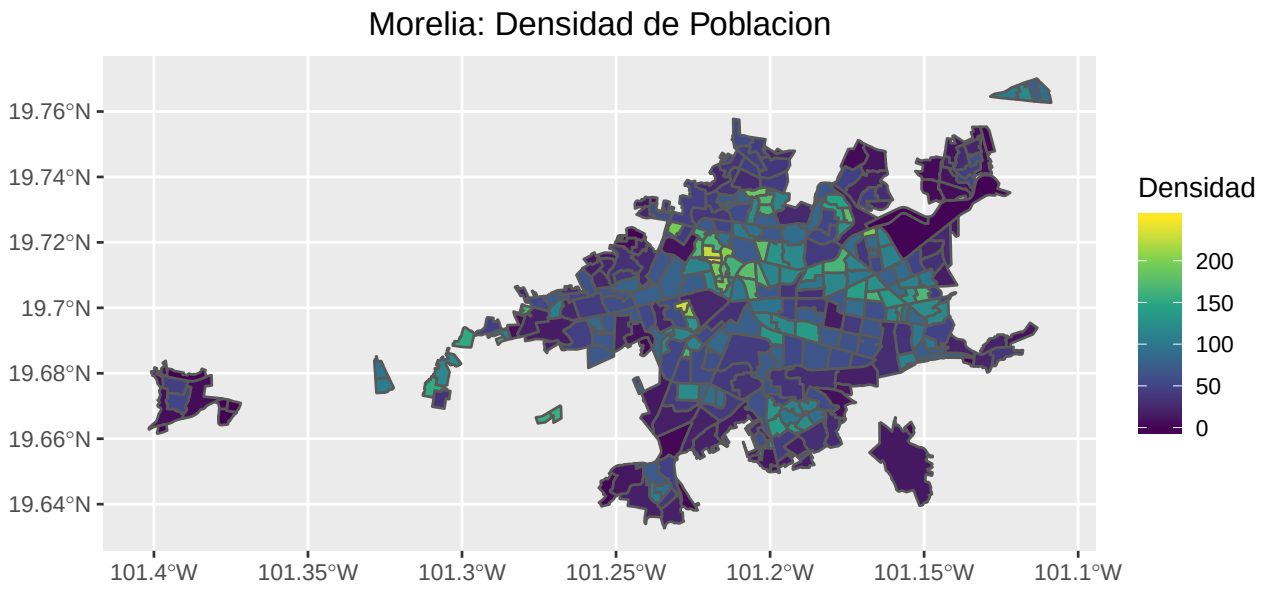
Como son datos espaciales, representan algun lugar en el espacio y muy seguramente tiene un sistema de referencia de coordenadas. Esto lo podemos verificar con `st_crs(densidad)`, que nos dice que tiene un sistema de referencia de coordenadas `EPSG 32614`, que es UTM zona 14 N. Lo queremos reproyectar en coordenadas geográficas, es decir, usar `EPSG 4326`. Esto lo hacemos con:

```
densidad <- st_transform(densidad, 4326)
```

Ahora usamos ggplot (y en particular `geom_sf`) para hacer un primer mapa de los datos espaciales con una paleta de colores que varíe de acuerdo a la densidad de población:

```
mapa1 <- ggplot() +
  geom_sf(data = densidad, aes(fill = densipob)) +
  ggtitle("Morelia: Densidad de Poblacion") +
```

```
  scale_fill_viridis_c(option = "D", name = "Densidad") +
  theme(
    plot.title = element_text(hjust = 0.5),
    plot.margin = unit(c(0.5, 0.5, 0.5, 0.5), "cm"))
mapa1
```



Ahora vamos a crear un mapa base de la zona de estudio. Usamos la función `get_map` y le decimos que use las coordenadas de la caja que contiene a los datos espaciales y así se evita tener que saber qué coordenadas pedir:

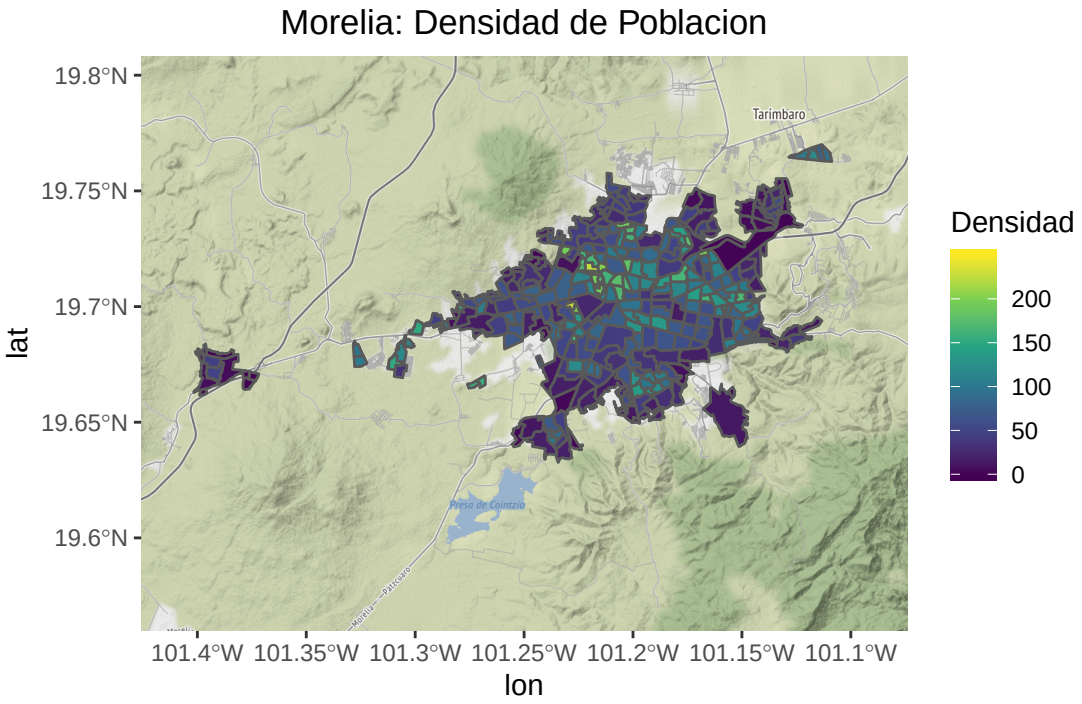
```
mapa_morelia <- get_map(location = matrix(st_bbox(densidad))[,1],
                        zoom = 12,
                        source = "stamen",
                        maptype = "terrain",
                        crop = FALSE)
#ggmap(mapa_morelia)
```

Se puede combinar la librería `ggmap` con la sintaxis de `ggplot` y podemos juntar el mapa base y los datos espaciales:

```
mapa2 = ggmap(mapa_morelia) +
  geom_sf(data = densidad, aes(fill = densipob), inherit.aes = FALSE)+
  ggtitle("Morelia: Densidad de Poblacion") +
  scale_fill_viridis_c(option = "D", name = "Densidad") +
  theme(
    plot.title = element_text(hjust = 0.5),
    plot.margin = unit(c(0.5, 0.5, 0.5, 0.5), "cm"))
```

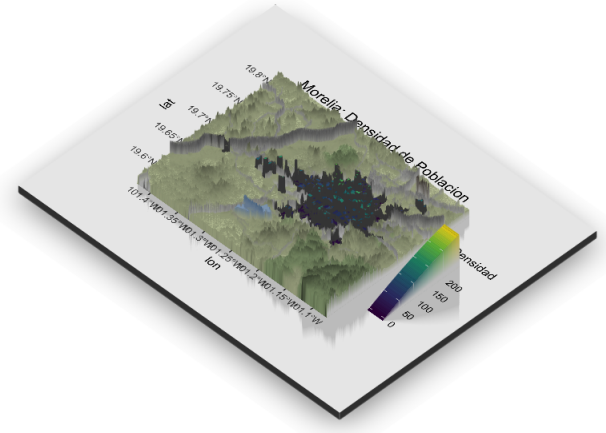
```
## Coordinate system already present. Adding new coordinate system, which will replace the existing one
```

```
mapa2
```



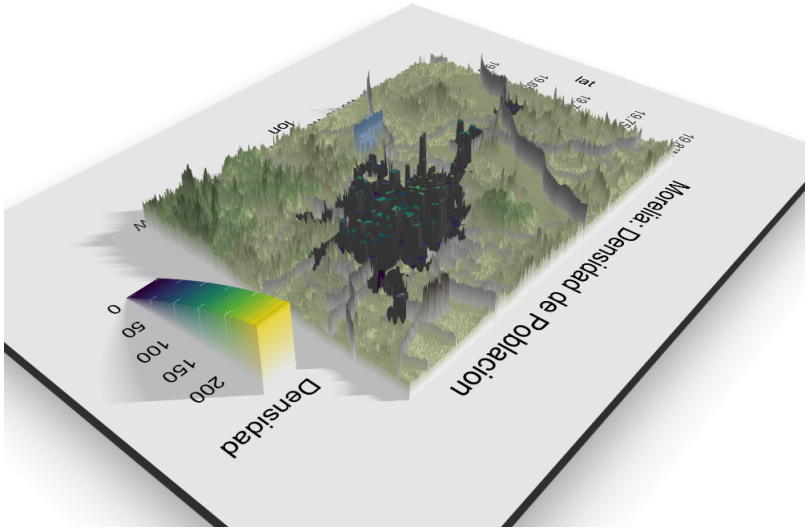
Ahora intentemos crear un mapa en 3D interactivo al usar la función `plot_gg` (paciencia... puede tardar un poco):

```
plot_gg(mapa2, width = 5, height = 4, windowsize = c(1000, 1000), multicore = TRUE)
```



Incluso puedes cambiar programáticamente cómo quieres que se vea tu mapa en 3D:

```
render_camera(fov = 70, zoom = 0.5, theta = 130, phi = 35)
```



Este mapa en 3D se ve muy mal porque al pasar de 2D a 3D se usaron *todos* los datos disponibles, tanto del mapa base como de los datos que nos interesan. Para arreglar esto, hay que hacer un truco: usar un mapa transparente para las elevaciones y ponerle encima el otro.

Vamos por partes. Primero usamos el mapa base y le extraemos sus atributos; luego creamos un nuevo objeto que sale de modificar la transparencia de ese mapa base y le pegamos los atributos originales:

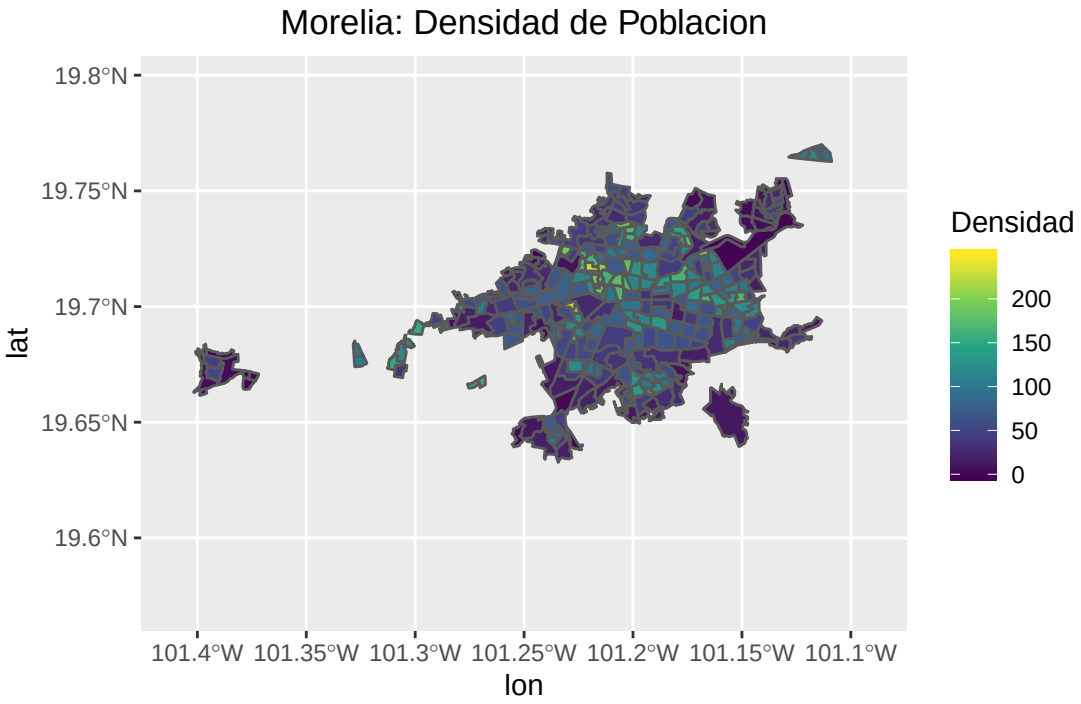
```
mapa_morelia_attributes <- attributes(mapa_morelia)
mapa_morelia_trans <- matrix(adjustcolor(mapa_morelia,
                                         alpha.f = 0),
                             nrow = nrow(mapa_morelia))
attributes(mapa_morelia_trans) <- mapa_morelia_attributes
```

Ahora podemos crear un objeto de tipo `ggplot` que contiene este mapa "transparente" y le agregamos una capa con las geometrías que nos interesan. El resultado es muy parecido al `mapa1`, pero garantizamos que tenga la extensión del `mapa2`:

```
extruidos <- ggmap(mapa_morelia_trans) +
  geom_sf(data = densidad, aes(fill = densipob), inherit.aes = FALSE) +
  ggtitle("Morelia: Densidad de Poblacion") +
  scale_fill_viridis_c(option = "D", name = "Densidad") +
  theme(
    plot.title = element_text(hjust = 0.5),
    plot.margin = unit(c(0.5, 0.5, 0.5, 0.5), "cm"))
```

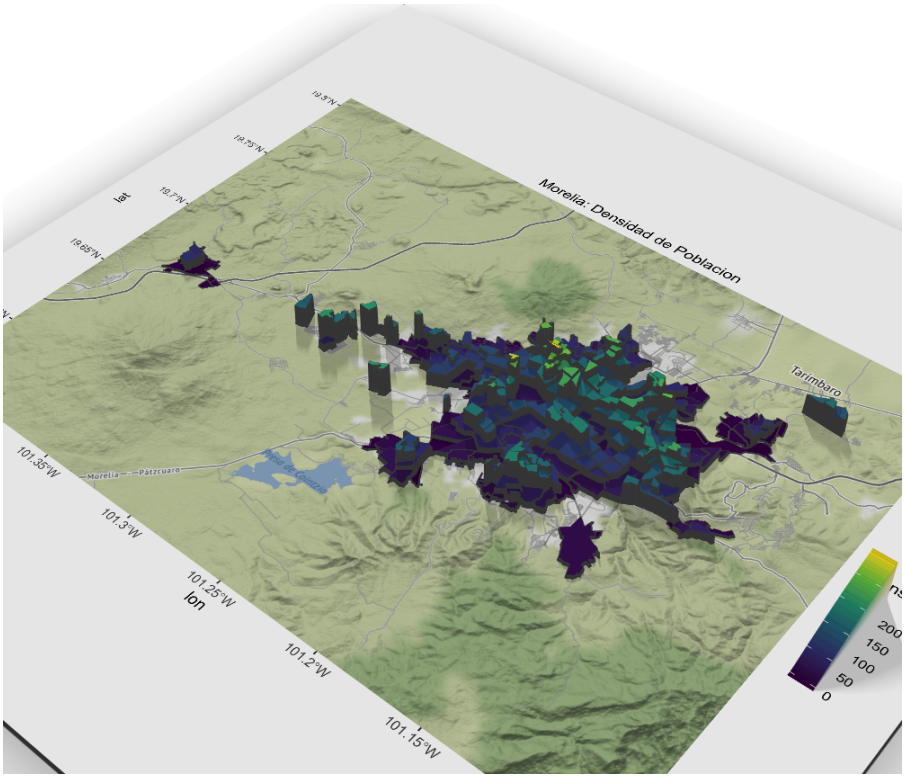
```
## Coordinate system already present. Adding new coordinate system, which will replace the existing one
```

```
extruidos
```



Y por último podemos generar el mapa base con polígonos extruidos. En este caso, el primer argumento de `plot_gg` es una lista de dos elementos: el primero es un `ggplot` de lo que se va a desplegar como base (en nuestro caso es `mapa2` que tiene el map base de *Stamen* y los polígonos tematizados) y el segundo es el `ggplot` que se usa para generar la superficie en 3D (solo los polígonos tematizados).

```
plot_gg(list(mapa2, extruidos),
        width = 10,
        height = 10,
        fov = 70,
        zoom = 0.25,
        windowsize = c(1000, 1000),
        multicore = TRUE)
```


Morelia- Densidad de Poblacion
Tarimbaro
Morelia
Pátzcuaro
lon
101.35°W
101.3°W
101.25°W
101.2°W
101.15°W
200
150
100
50
0