

1. Empezando con OSM

Lo primero que deberemos tener en cuenta son las librerías que necesitaremos cargar o instalar para obtener los datos, trabajar con ellos y mostrarlos de forma visual. Las librerías que han de ser cargadas son las siguientes:

- **Leaflet**: es una librería JavaScript open-source ampliamente utilizada para la publicación de mapas en la web.
- **Osmdata**: contiene los datos necesarios para trabajar con los datos de OpenStreetMap.
- **Tidyverse**: es un conjunto de paquetes en R diseñados para ciencia de datos.
- **Sf**: para el uso de los Data-Frames
- **Ggmap**: Para la visualización de los mapas.
- **Ggplot2**: Para la visualización de los datos y características.

Será necesario usar `library("nombre_de_la_libreria")`

En caso de no tenerlas descargadas y tener que instalarlas: `install.packages("librería")`.

2. ¿Qué datos puedo analizar?

Para ello, podemos listar las características que nos ofrece la librería.

Usaremos `available_features()`

Cuando lo hemos ejecutado, obtendremos una lista con todos los valores sobre los que podemos realizar búsquedas en nuestros mapas. Algunos de ellos, incorporan subtipos.

Para averiguarlo, usaremos: `available_tags("nombre_del_feature")`

Por ejemplo, si ponemos usamos "shop", este sería el resultado:

> <code>available_tags("shop")</code>	
[1] "agrarian"	"alcohol"
[4] "antiques"	"appliance"
[7] "atv"	"baby_goods"
[10] "bakery"	"bathroom_furnishing"
[13] "bed"	"beverages"
[16] "boat"	"bookmaker"
[19] "boutique"	"brewing_supplies"
[22] "camera"	"candles"
[25] "car"	"car_parts"
[28] "caravan"	"carpet"
[31] "cheese"	"chemist"
[34] "clothes"	"coffee"
[37] "computer"	"confectionery"
[40] "copyshop"	"cosmetics"
[43] "curtain"	"dairy"
[46] "department_store"	"doityourself"
[49] "dry_cleaning"	"e-cigarette"
[52] "electronics"	"energy"
[55] "fabric"	"farm"
[58] "fashion_accessories"	"fireplace"
[61] "flooring"	"florist"
[64] "frozen_food"	"fuel"
[67] "furniture"	"games"
[70] "garden_furniture"	"gas"
[73] "gift"	"glazier"
[76] "greengrocer"	"groundkeeping"
[79] "hairdresser_supply"	"hardware"
[82] "hearing_aids"	"herbalist"
[85] "household_linen"	"houseware"
[88] "ice_cream"	"interior_decoration"
	"anime"
	"art"
	"bag"
	"beauty"
	"bicycle"
	"books"
	"butcher"
	"cannabis"
	"car_repair"
	"charity"
	"chocolate"
	"collector"
	"convenience"
	"craft"
	"deli"
	"doors"
	"electrical"
	"erotic"
	"fashion"
	"fishing"
	"frame"
	"funeral_directors"
	"garden_centre"
	"general"
	"golf"
	"hairdresser"
	"health_food"
	"hifi"
	"hunting"
	"jet ski"

Una vez obtengamos estos listados y sepamos qué es lo que queremos buscar, podemos seleccionar los datos.

3. Construcción de la consulta

Como ya hemos visto durante las prácticas de la asignatura, vamos a guardar en variables los datos que luego vamos a representar. Estos procesos se van a realizar mediante el uso de una “query”.

Primeramente, será necesario obtener las coordenadas desde donde queremos operar. Para ello, realizamos la siguiente consulta:

```
getbb("Nombre de lo que queremos consultar")
```

El idioma en que se realizarán las consultas será en inglés. Por tanto, podemos buscar por Spain, Germany, Italy... o bien por comunidades autónomas dentro del país: Granada Spain, Hamburg Germany, Rome Italy...

Yo voy a empezar con un ejemplo de España. Lo primero será calcular sus coordenadas, para ello lo realizaremos de la siguiente forma:

> getbb("Spain")
min max
x -83.55794 -83.51794
y 30.84464 30.88464

Sin embargo, para países esta forma no es del todo fiable y da algunos errores. Para ello, visitaremos la página <https://gist.github.com/graydon/11198540> en donde encontraremos las coordenadas asociadas a nuestro país.

Y por la línea de comandos guardaremos el valor de las coordenadas:

```
coordenadas <- c(-9.39288367353, 35.946850084, 3.03948408368, 43.7483377142)
```

Ahora, tendremos que construir la consulta. En este caso, vamos a visualizar los supermercados Eroski que hay en España.

q <- coordenadas %>%
opq() %>%
add_osm_feature("name", "Eroski") %>%
add_osm_feature("shop", "supermarket")

Para construir la consulta se hace uso de la función *opq()* y se van añadiendo atributos con *add_osm_features()*. Para concatenar las distintas funciones se usa el operador *%>%*. En concreto dentro de las tiendas (shop) seleccionamos las que son de tipo ‘*Eroski*’. Esto se almacena en la variable que hemos llamado q.

El siguiente paso será enviar la consulta al sistema osm:

eroskis <- osmdata_sf(q)

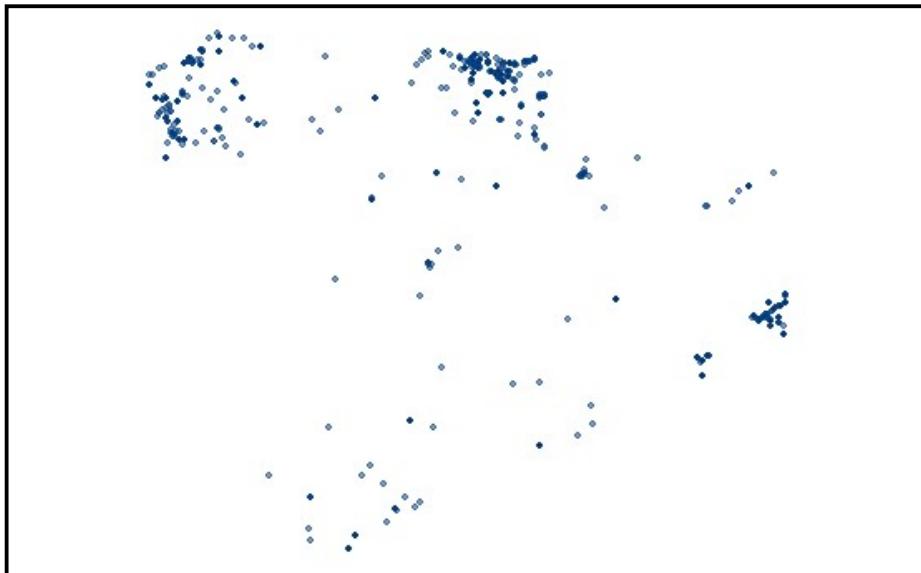
El valor del resultado lo guardaremos en otra variable, que en concreto se llamará “*eroskis*” para ser un nombre intuitivo. La función a la que se llama es *osmdata_sf* que recibe como parámetro la consulta creada.

Finalmente pasaremos al proceso de visualizar el resultado.

```
ggplot(eroskis$osm_points)+  
  geom_sf(colour = "#08519c",  
          fill = "#08306b",  
          alpha = .5,  
          size = 1,  
          shape = 21)+  
  theme_void()
```

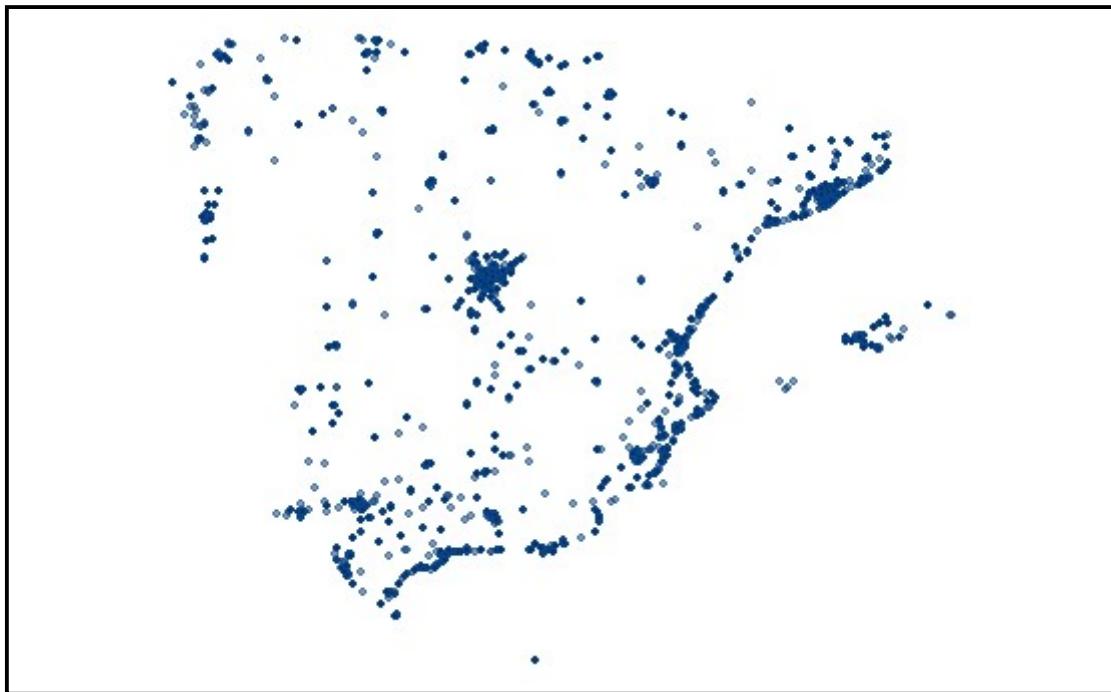
Se usará para la visualización la función *ggplot*, que se le pasará por parámetro la variable anterior (*eroskis*) y a la que podemos añadirle características para mejorar la visibilidad. Para ello, usaremos *geom_sf* y como parámetros *colour* con el color del contorno, *fill* con el color del relleno, *Alpha* que es para obtener distintas gamas del color y poder distinguir valores superpuestos, *size* para el tamaño de los puntos y *shape* para la forma que queremos (El 21 corresponde con el circulo, el 1 con triángulos el 10 con cuadrados...), por último, se añadirá la función *theme_void()* para realizar el volcado por pantalla.

El resultado que lo podemos exportar con la opción de Export, será el siguiente:



Localización de los Eroskis en España

Si ejecutamos de nuevo todo, pero en vez de ‘Eroski’ ponemos ‘Mercadona’, este será el resultado:



Localización de los Mercadonas en España

4. Consulta sobre una comunidad

Una vez realizadas las pruebas sobre un país, voy a realizarlo sobre una comunidad autónoma. En mi caso cogeré Almería por ser la ciudad de mi familia materna. Para este caso, vamos a analizar la red de carreteras que se encuentran en Almería.

```
getbb("Almeria Spain")
min <- c(-3.037, 35.937)
max <- c(-2.202, 37.000)
almeria_df <- as.matrix(data.frame(min, max))
row.names(almeria_df) <- c("x", "y")
```

Seleccionamos las coordenadas y las almacenamos en una variable.

Creamos la consulta con la red de carreteras

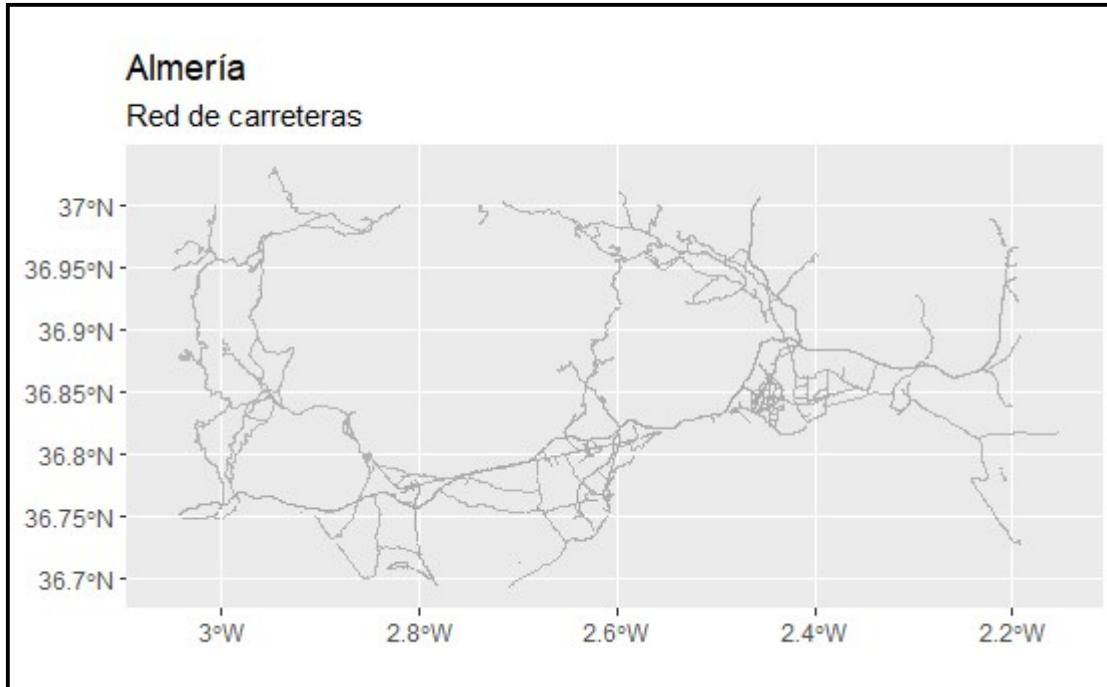
```
carreteras <- getbb("Almeria Spain")%>%
  opq()%>%
  add_osm_feature(key = "highway",
                   value = c("road", "motorway", "primary",
                            "secondary", "tertiary")) %>%
  osmdata_sf()
```

Generamos el mapa:

```
ggplot() +
  geom_sf(data = carreteras$osm_lines,
          inherit.aes = FALSE,
          color = "darkgrey",
          size = .4,
          alpha = .8) +
  labs(title = "Almería",
       subtitle = "Red de carreteras")
```

En este caso, vamos a incluir la función labs, para introducir el title y el subtitle al igual que el parámetro inherit.aes=FALSE, para obtener los ejes de coordenadas.

El resultado será el siguiente:



Visto el mapa de esta forma, queda un poco aburrido y no tiene mucho sentido. Para ello vamos a obtener las calles de Almería y vamos a superponer los mapas. Sería algo similar a superponer capas en Qgis. Vamos a obtener las calles de Almería:

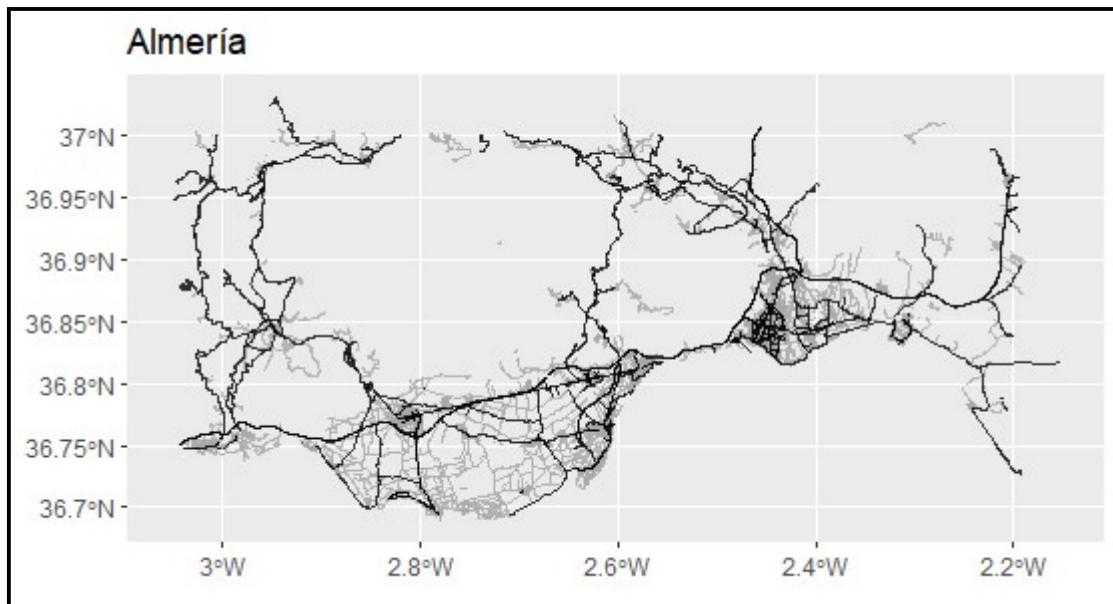
```
calles <- getbb("Almería Spain")%>%
  opq()%>%
  add_osm_feature(key = "highway",
                  value = c("residential", "living_street",
                           "unclassified",
                           "service", "footway")) %>%
  osmdata_sf()
```

Una vez obtenidas, las vamos a visualizar superpuestas con las carreteras:

```
ggplot() +
  geom_sf(data = calles$osm_lines,
          inherit.aes = FALSE,
          color = "darkgrey",
          size = .2,
          alpha = .8) +
  geom_sf(data = carreteras$osm_lines,
          inherit.aes = FALSE,
          color = "black",
          size = .2,
          alpha = .8) +
  labs(title = "Almería")
```

Lo único que tenemos que hacer es añadir otra función geom_sf con los valores de las calles.

El resultado final será:



Si queremos, podemos añadir algo de diseño añadiendo al final la función theme

```
ggplot() +
  geom_sf(data = calles$osm_lines,
          inherit.aes = FALSE,
          color = "orange",
          size = .1,
          alpha = .8) +
  geom_sf(data = carreteras$osm_lines,
          inherit.aes = FALSE,
          color = "grey40",
          size = .2,
          alpha = .8) +
  theme_void() +
  labs(title = "Almería") +
  theme(plot.background = element_rect(fill = "black"),
        plot.title = element_text(colour = "white"))
```

Le indicamos que el fondo sea negro, lo que queremos destacar en naranja, las calles en gris y las letras en blanco.

El resultado obtenido queda más visual de cara al cliente:



Aquí podemos añadir la capa redHidrográfica y visualizar los ríos que se encuentran en Almería.

Muestro como se crearía la consulta de los ríos:

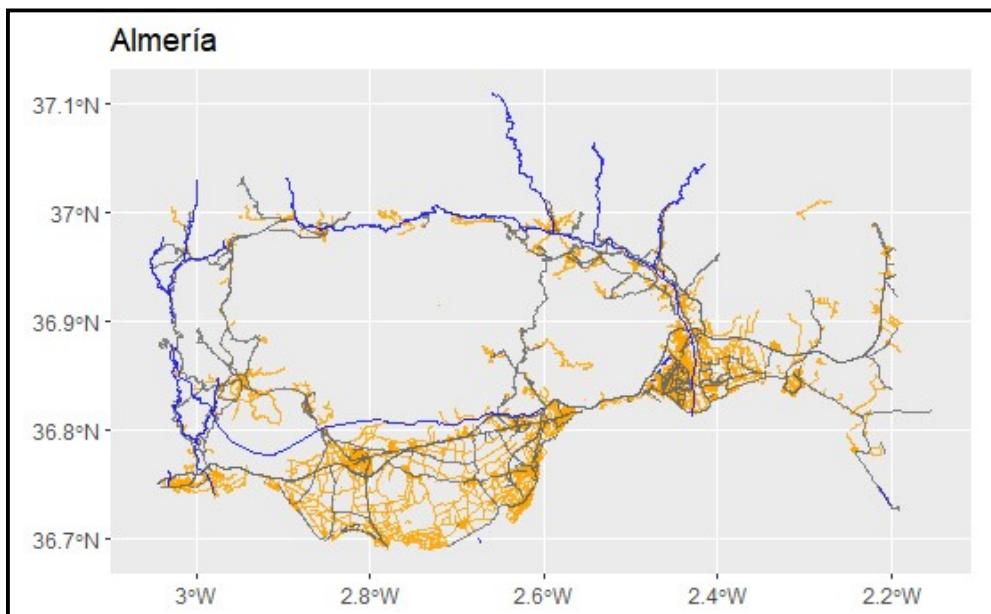
```
rios <- getbb("Almeria Spain")%>%
  opq()%>%
  add_osm_feature(key = "waterway",
                  value = c("river", "canal")) %>%
  osmdata_sf()
```

Crear la capa ríos

```
ggplot() +
  geom_sf(data = calles$osm_lines,
          inherit.aes = FALSE,
          color = "orange",
          size = .1,
          alpha = .8) +
  geom_sf(data = carreteras$osm_lines,
          inherit.aes = FALSE,
          color = "grey40",
          size = .2,
          alpha = .8) +
  geom_sf(data = rios$osm_lines,
          inherit.aes = FALSE,
          color = "blue",
          size = .2,
          alpha = .8) +
  labs(title = "Almeria")
```

Visualizar las capas

El resultado final:



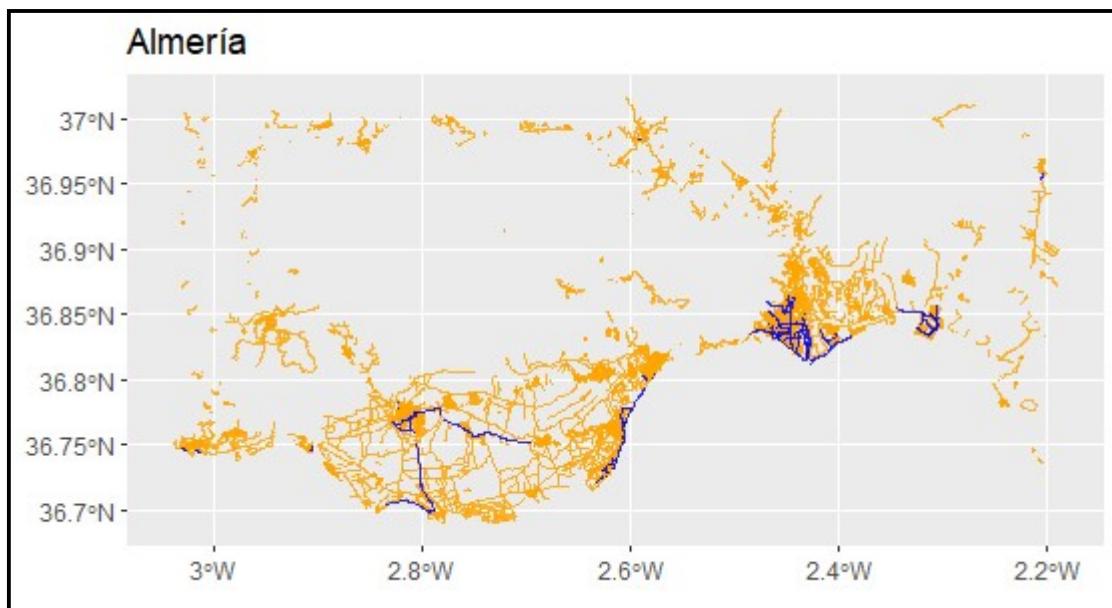
Otra cosa interesante que nos permite visualizar son los carriles bici. Lo muestro debido a que estos últimos años está en auge y así podemos ver cuanto abarca este medio de transporte.

```
ggplot() +
  geom_sf(data = calles$osm_lines,
          inherit.aes = FALSE,
          color = "orange",
          size = .1,
          alpha = .8) +
  geom_sf(data = bici$osm_lines,
          inherit.aes = FALSE,
          color = "blue",
          size = .2,
          alpha = .8) +
  labs(title = "Almeria")
```

```
bici <- getbb("Almeria Spain")%>%
  opq()%>%
  add_osm_feature(key = "highway",
                  value = c("cycleway")) %>%
  osmdata_sf()

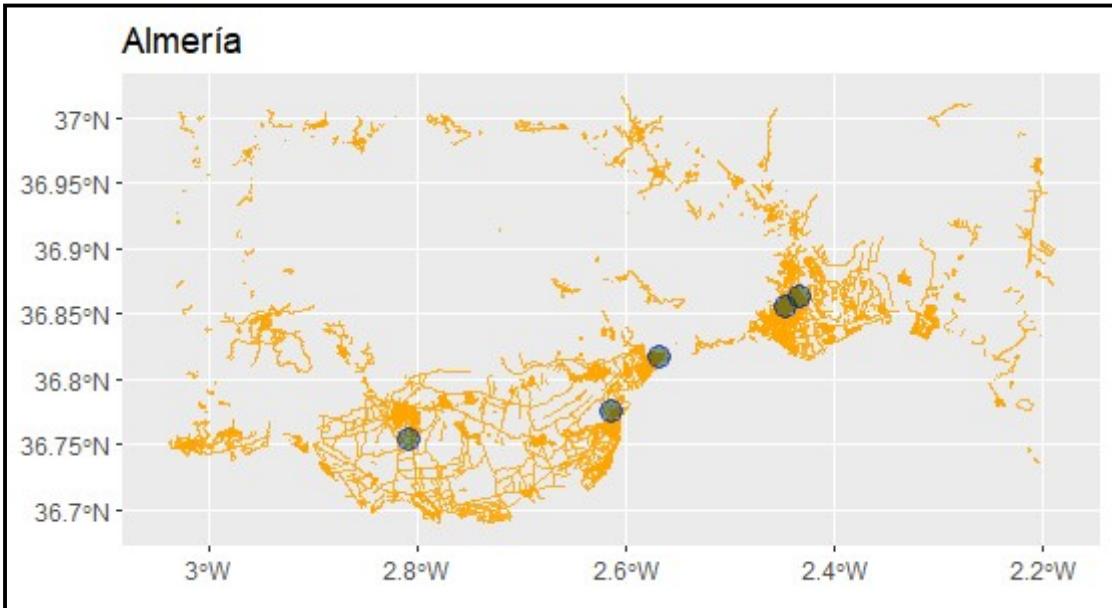
ggplot() +
  geom_sf(data = calles$osm_lines,
          inherit.aes = FALSE,
          color = "orange",
          size = .1,
          alpha = .8) +
  geom_sf(data = bici$osm_lines,
          inherit.aes = FALSE,
          color = "orange",
          size = .1,
          alpha = .8) +
  geom_sf(data = bici$osm_lines,
          inherit.aes = FALSE,
          color = "blue",
          size = .2,
          alpha = .8) +
  labs(title = "Almeria")
```

Carril bici:



La última funcionalidad que vamos a visualizar con este mapa es la localización en forma de puntos de algunos establecimientos en concreto. Ya lo hemos hecho en la primera parte, pero ahora lo vamos a visualizar para un mapa más pequeño. En concreto veremos dentro de la etiqueta *amenity*, cuantos cines hay en Almería.

Para ello, creamos la capa cines:



Todos estos datos, los estamos mostrando en un formato liso. Sin embargo, podemos obtenerlos sobre mapas reales. Para ello, será necesario crearnos un mapa virtual de la zona, sin embargo, esto no es aplicable para el país.

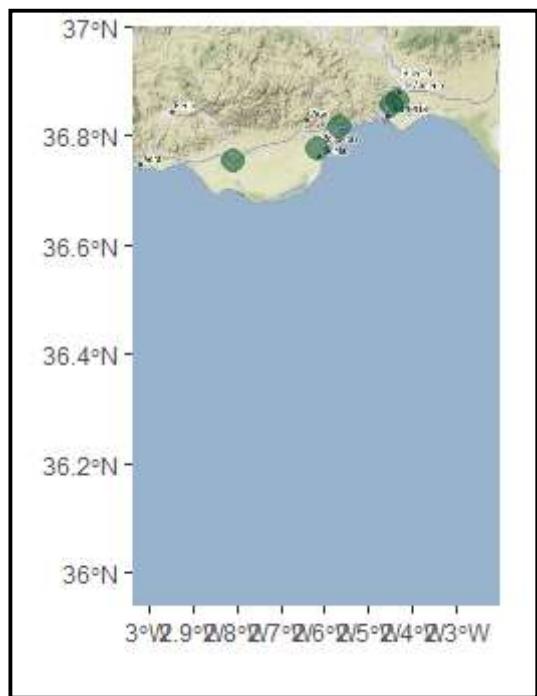
Para crear el mapa usaremos la siguiente sentencia:

```
mad_map <- get_map(getbb("Ciudad País"), maptype = "toner-background")
```

Y para mostrar la información usaremos la función ggmap y el resto será igual.

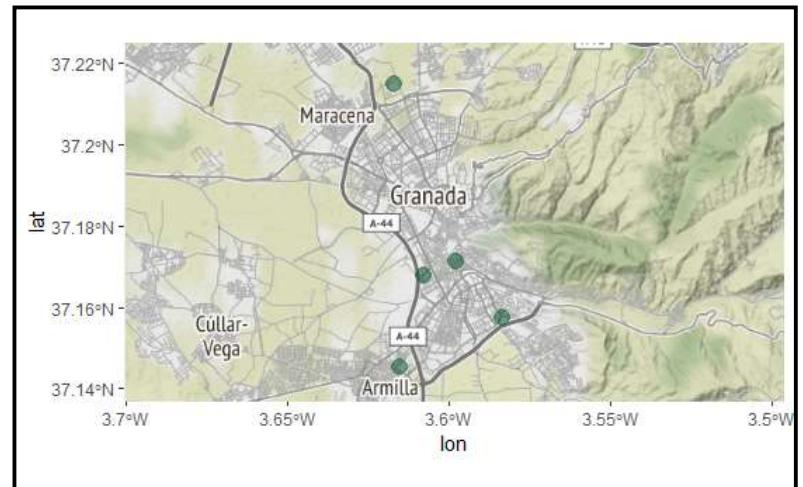
```
mad_map <- get_map(getbb("Almería Spain"), maptype = "toner-background")
ggmap(mad_map) +
  geom_sf(data = cinema$osm_points,
          inherit.aes = FALSE,
          colour = "#238443",
          fill = "#004529",
          alpha = .5,
          size = 4,
          shape = 21)
```

El resultado será el siguiente:



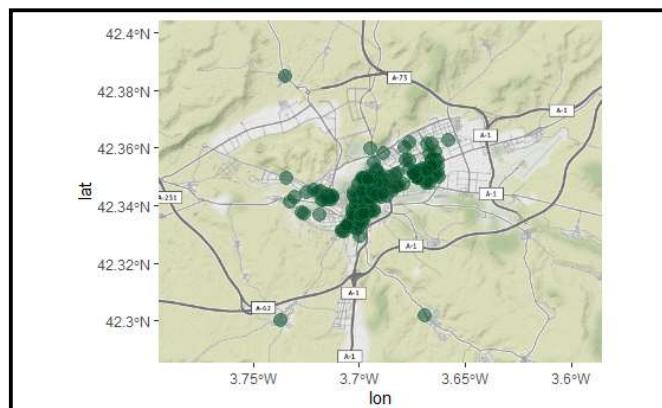
Resultado de los cines en Almería

Probamos el mismo resultado en Granada:

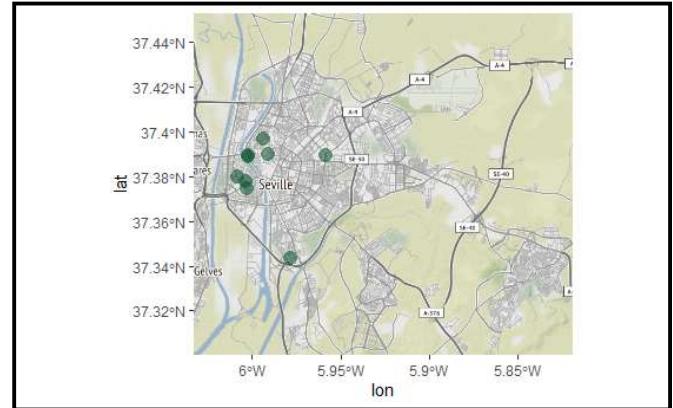


Resultados de cines en Granada

Podemos realizar lo mismo, cambiando las ciudades y los establecimientos:

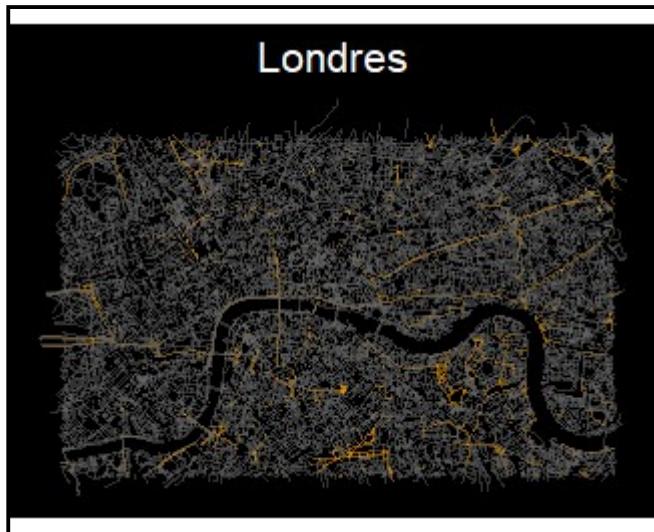


Peluquerías en Burgos

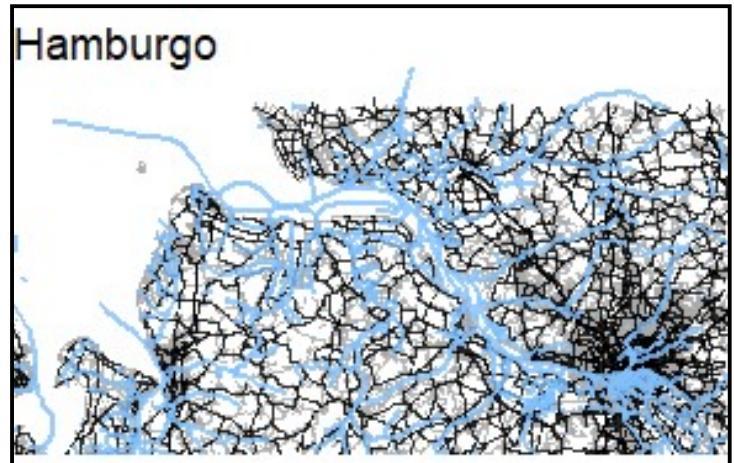


Tiendas de tatuajes en Sevilla

Ejemplos con otros Países



Carril bici en Londres



Ríos en Hamburgo

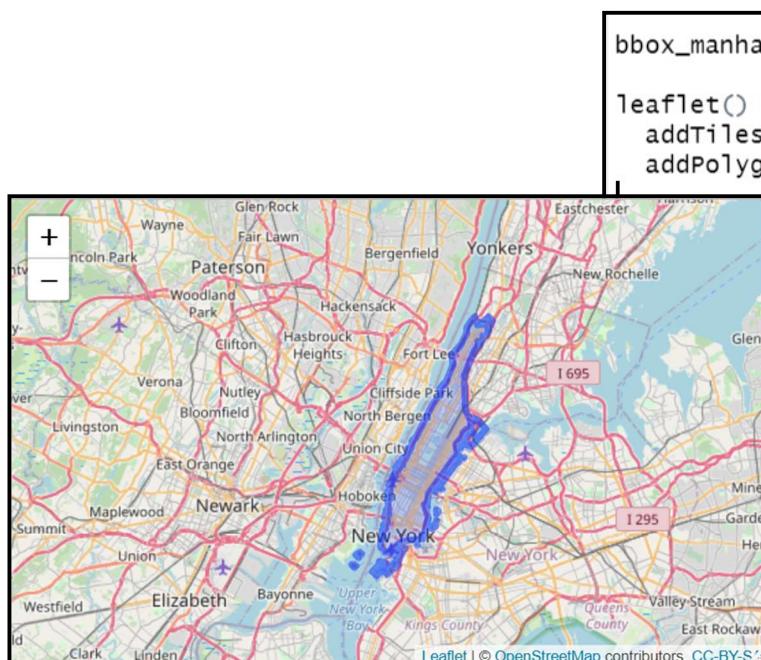
Otras utilidades de OSM

Podemos seleccionar en un mapa una parte que queramos estudiar y luego extraerla. Para realizar este estudio, vamos a utilizar Manhattan y un archivo GeoJson.

Lo cargamos de esta forma:

```
bbox_poly_manhattan <- st_read("https://data.cityofnewyork.us/api/geospatial/tqmj-j8zm?method=export&format=GeoJSON") %>%  
filter(boro_name == "Manhattan")
```

Almacenamos el polígono en una variable y lo mostramos:



```
bbox_manhattan <- st_bbox(bbox_poly_manhattan)  
leaflet() %>%  
addTiles() %>%  
addPolygons(data = bbox_poly_manhattan)
```

Vemos como Manhattan queda destacada del resto.

Ahora obtendremos sólo el mapa de Manhattan, para ello habrá que realizar la intersección entre el polígono anterior y sus calles.

```
callesMan <- opq(bbox_manhattan) %>%
  add_osm_feature(key = "highway")

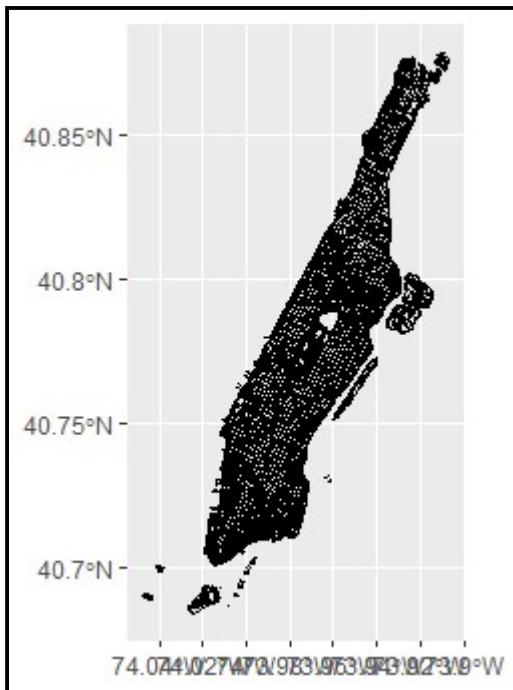
callesMan <- callesMan %>%
  osmdata_sf()
```

```
callesMan <- callesMan$osm_lines

callesMan <- st_intersection(callesMan, bbox_poly_manhattan)

ggplot() +
  geom_sf(data = callesMan)
```

El resultado será:



Otra utilidad que vamos a ver es establecer un rango de colores. En este caso, según el número de carriles, lo vamos a visualizar de un color diferente. Para ello, deberemos hacer varios pasos antes:

```
callesManla <- callesMan %>%
  mutate(lanes = as.numeric(lanes),
        lanes = ifelse(is.na(lanes), 1, as.numeric(lanes)))

callesManla <- callesManla %>%
  mutate(color=case_when(lanes=="1" ~ "midnightblue",
                        lanes=="2" ~ "darkorange1",
                        lanes=="3" ~ "red4",
                        lanes=="4" ~ "brown4",
                        lanes=="5" ~ "darkmagenta",
                        lanes=="6" ~ "yellow3",
                        lanes=="7" ~ "green4",
                        lanes== "NA" ~ "aliceblue"))

ggplot() +
  geom_sf(data = bbox_poly_manhattan, aes(), color="grey80", size = 0.4) +
  geom_sf(data = callesManla, aes(fill = lanes)) +
  geom_sf(data = callesManla, aes(color = lanes), size = 0.5) +
  scale_color_identity()
  theme_void()
  labs(title = "Manhattan - NYC",
       subtitle = "Vías de circulación",
       fill = "Cantidad de carriles",
       caption = "Fuente: OpenStreetMap")
```

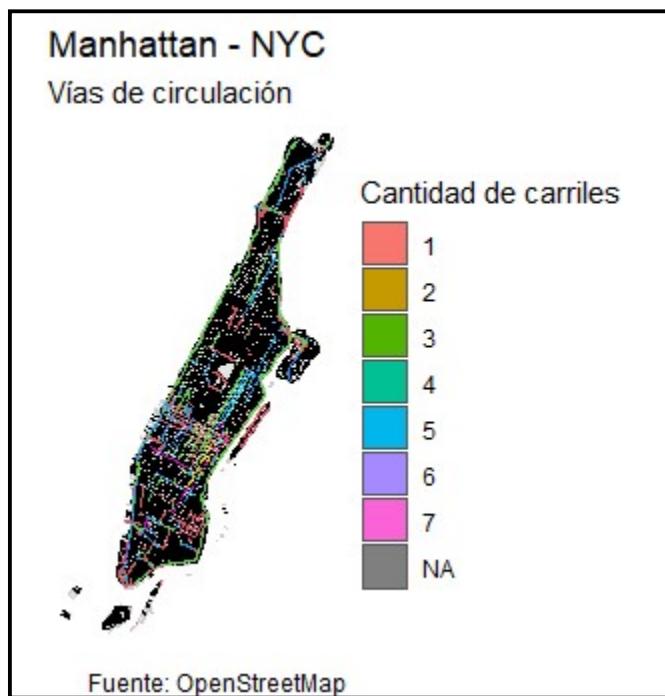
Primero, convertiremos los valores de las calles en números y los valores perdidos en NA.

Luego establecemos los colores del 1 al 7 y también para el NA.

Finalmente, tendremos que añadir a la visualización:

Las calles con el formato original, para ver el fondo negro, el calles con el formato que le hemos dado y la función `scale_color_identity()` para poder

hacer de forma correcta la visualización de los diferentes colores:



Por último, visualizaremos los cafés que tienen en Manhattan. Para ello, aparte de extraerlos como hemos hecho anteriormente, habrá que realizar de nuevo una intersección entre los puntos de los cafés y el polígono inicial.

Obtenemos los cafés, lo convertimos en un objeto *osmdata_sf*, lo ponemos en formato de puntos, realizamos la intersección y lo visualizamos como hemos explicado en los primeros puntos.

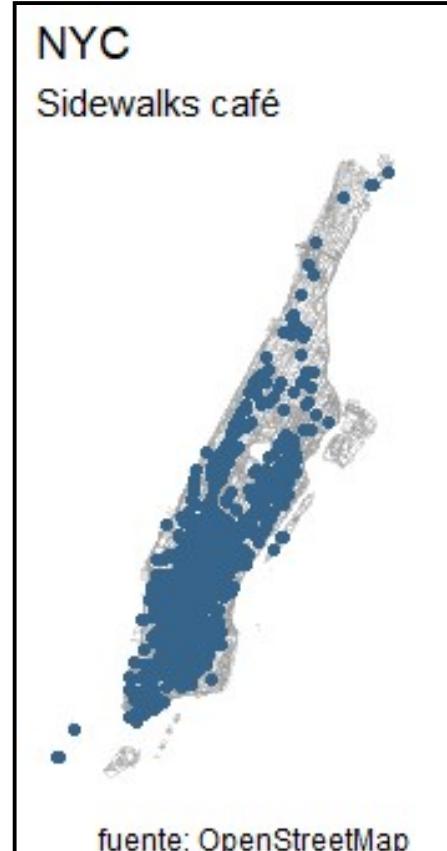
```
cafeMan <- opq(bbox_manhattan) %>%
  add_osm_feature(key = "amenity", value = "cafe")

cafeMan <- cafeMan %>%
  osmdata_sf()

cafeMan <- cafeMan$osm_points

cafeMan <- st_intersection(cafeMan, bbox_poly_manhattan)|

ggplot() +
  geom_sf(data = callesMan,
         color = "gray60", alpha = 0.3) +
  geom_sf(data = cafeMan, color = "steelblue4") +
  theme_void() +
  labs(title = "NYC",
       subtitle = "Café",
       caption = "fuente: OpenStreetMap")
```



Bibliografía

<https://cran.r-project.org/web/packages/osmdata/vignettes/osmdata.html>

<https://cran.r-project.org/web/packages/osmdata/index.html>

<https://dominicroye.github.io/en/2018/accessing-openstreetmap-data-with-r/>

<https://github.com/ropensci/osmdata>

<https://rpubs.com/Eugegonz/519666>

<https://rpubs.com/HAVB/osm>