Curso Internacional de Desagregación de Estimaciones en Áreas Pequeñas usando R

Realización de mapas para indicadores desagregados geográficamente usando R

División de Estadísticas
Comisión Económica para América Latina y el Caribe

2020

Datos cartográficos

Introducción

- Los mapas son una herramienta gráfica poderosa para la visualización de datos.

Introducción

- Los mapas son una herramienta gráfica poderosa para la visualización de datos.
- Para indicadores sociales-demográficos estos son una gran referencia visual para desagregaciones a nivel País, región, departamento, provincia, distrito, municipio, comuna, etc.

Introducción

- Los mapas son una herramienta gráfica poderosa para la visualización de datos.
- Para indicadores sociales-demográficos estos son una gran referencia visual para desagregaciones a nivel País, región, departamento, provincia, distrito, municipio, comuna, etc.
- El software de código libre utilizado para análisis estadístico `R` posee un sin fin de métodos de programación para representar dichos mapas.

Datos cartográficos

- Para graficar mapas es necesario contar con información geoespacial, datos que contienen las coordenadas o delimitaciones geográficas de determinado país o región.

Datos cartográficos

- Para graficar mapas es necesario contar con información geoespacial, datos que contienen las coordenadas o delimitaciones geográficas de determinado país o región.
- Sitios web como http://www.diva-gis.org/gdata ofrecen de manera gratuita bases de datos o `shapes` que contienen los vectores asociados a las geografías correspondientes.

Datos cartográficos

- Dichos conjuntos de datos poseen observaciones sobre la longitud y latitud lo cuál permite graficar en R un conjunto de puntos cuya unión en el gráfico formarán las formas los polígonos que dan forma a las áreas geográficas.

Datos cartográficos

- Dichos conjuntos de datos poseen observaciones sobre la longitud y latitud lo cuál permite graficar en R un conjunto de puntos cuya unión en el gráfico formarán las formas los polígonos que dan forma a las áreas geográficas.
- Como ejemplo, se utilizarán los resultados obtenidos mediante la metodología SAE en Perú a nivel provincia para el indicador ODS 3.7.1 o D.7 del consenso de Montevideo (Mujeres unidas en edad fértil que utilizan métodos modernos).

Datos cartográficos

- La base de datos contiene el nombre del departamento, el código `ubigeo` para las circunscripciones territoriales (a nivel provincia) y la estimación puntual del indicador para dicha provincia.

id	departamento	ubigeo	D7
0	Amazonas	0101	0.7674031
1	Amazonas	0102	0.7320016
2	Amazonas	0103	0.7494394
3	Amazonas	0104	0.3430710
4	Amazonas	0105	0.6352737
5	Amazonas	0106	0.8246836

Datos cartográficos

- La librería `rgdal` de `R` nos permite leer la información geoespacial en formato `.shp` contenida en la carpeta llamada `PER_adm`.

Datos cartográficos

- La librería `rgdal` de `R` nos permite leer la información geoespacial en formato `.shp` contenida en la carpeta llamada `PER_adm`.
- `ohsPER2` contiene la información geoespacial necesaria para mapear los polígonos a nivel provincia en Perú.

Datos cartográficos

- Perú se divide administrativamente en Departamentos, Provincias y distritos. La información anterior está construida para la segunda desagregación geográfica (196 provincias).

```
library(rgdal)
ohsPER2 <- readOGR(".../PER_adm/provincias/PROVINCIAS.shp")
```

Datos cartográficos

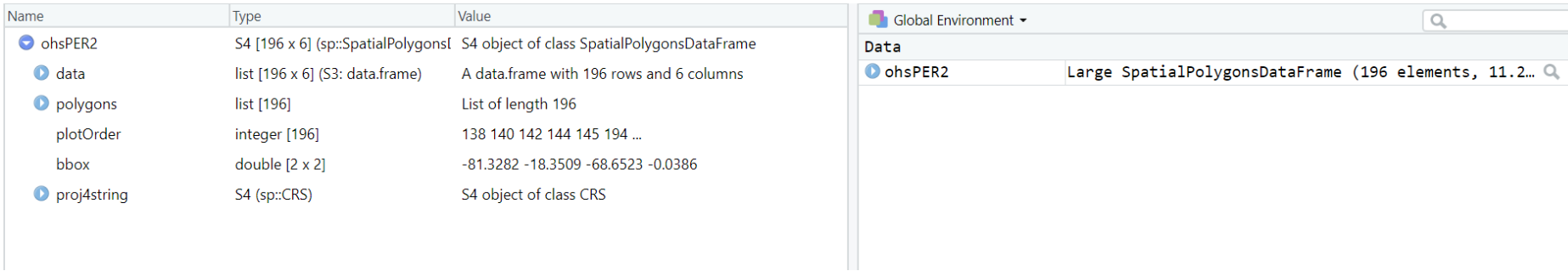


Figure 1: archivo .shp

Mapas con ggplot2

Librería ggplot

- `ggplot2` es una potente librería gráfica que permite crear mediante códigos computacionales distintos entornos gráficos en `R`.

Librería ggplot

- `ggplot2` es una potente librería gráfica que permite crear mediante códigos computacionales distintos entornos gráficos en `R`.
- Mediante `geom_polygon()` es posible utilizar un conjunto de vectores de un archivo `.shp` para graficar formas poligonales que generan el mapa de una determinada división administrativas de un país.

Librerías requeridas

Las librerías necesarias se muestran en la siguiente sintaxis.

```
library(dplyr)
library(tidyverse)
library(magrittr)
library(sp)
library(RColorBrewer)
library(ggplot2)
library(maptools)
library(scales)
library(gridExtra)
library(grid)
library(sf)
library(rgdal)
```

Librerías requeridas

- Las librerías `dplyr, tidyverse y magritr` se utilizan para realizar manejo de bases de datos.

Librerías requeridas

- Las librerías `dplyr, tidyverse y magritr` se utilizan para realizar manejo de bases de datos.
- `RColorBrewer` permite utilizar funciones para explorar las grillas de colores y paletas que dispone `R`.

Librerías requeridas

- Las librerías `dplyr, tidyverse y magritr` se utilizan para realizar manejo de bases de datos.
- `RColorBrewer` permite utilizar funciones para explorar las grillas de colores y paletas que dispone `R`.
- `sp` se utiliza para crear las `data.frame` a partir de la información geoespacial y poder realizar los emparejamientos necesarios con los datos que se desean graficar.

Librerías requeridas

- Las librerías `dplyr, tidyverse y magritr` se utilizan para realizar manejo de bases de datos.
- `RColorBrewer` permite utilizar funciones para explorar las grillas de colores y paletas que dispone `R`.
- `sp` se utiliza para crear las `data.frame` a partir de la información geoespacial y poder realizar los emparejamientos necesarios con los datos que se desean graficar.
- Las librerías `grid` y `gridExtra` permiten la unificación de distintos mapas en una sola grilla.

Librerías requeridas

- Una vez cargado el archivo `.shp` en el entorno de `R`, se utiliza la función `fortify()` para convertir la lista de información geoespacial en una `data.frame` con la cuál se realiza el emparejamiento mediante la variable `id`.

Librerías requeridas

- Una vez cargado el archivo `.shp` en el entorno de `R`, se utiliza la función `fortify()` para convertir la lista de información geoespacial en una `data.frame` con la cuál se realiza el emparejamiento mediante la variable `id`.
- Se debe tener en cuenta como está formado el id y la concordancia que tiene con la información que se dispone.

Librerías requeridas

- Una vez cargado el archivo `.shp` en el entorno de `R`, se utiliza la función `fortify()` para convertir la lista de información geoespacial en una `data.frame` con la cuál se realiza el emparejamiento mediante la variable `id`.
- Se debe tener en cuenta como está formado el id y la concordancia que tiene con la información que se dispone.
- Si la información `.shp` proviene de fuentes oficiales no se debería tener problemas al momento de la unión, información no oficial puede estar desactualizada u ordenada de otra forma.

Librerías requeridas

- En el caso de Perú, las provincias está ordenadas según la circunscripciones territoriales del INEI `Ubigeo`. Las provincias se ordenan por orden alfabético de departamento seguido de la capital del departamento, luego, se ordenan alfabéticamente las provincias correspondientes del departamento.

Librerías requeridas

- En el caso de Perú, las provincias está ordenadas según la circunscripciones territoriales del INEI `Ubigeo`. Las provincias se ordenan por orden alfabético de departamento seguido de la capital del departamento, luego, se ordenan alfabéticamente las provincias correspondientes del departamento.
- En este caso `id = 0` corresponde a Chacapoyas capital de Amazonas el primer departamento ordenado en orden alfabético. Los siguientes 6 id corresponden a las provincias que se encuentran en el departamento de Amazonas ordenados de manera alfabética.

Creando mapas en R con ggplot2

- Es necesario leer los *layers* del mapa. Perú se divide en 24 departamentos y una provincia constitucional que a su vez están constituidas por 196 provincias. Esta forma viene dada por el archivo `PER_adm2.shp`.

```
ohsPERI2 <- fortify(ohsPER2)
shapes <- ohsPERI2 %>% merge(D7, by = "id")
```

Creando mapas en R con ggplot2

- Se debe combinar estos datos con la variable de interés.

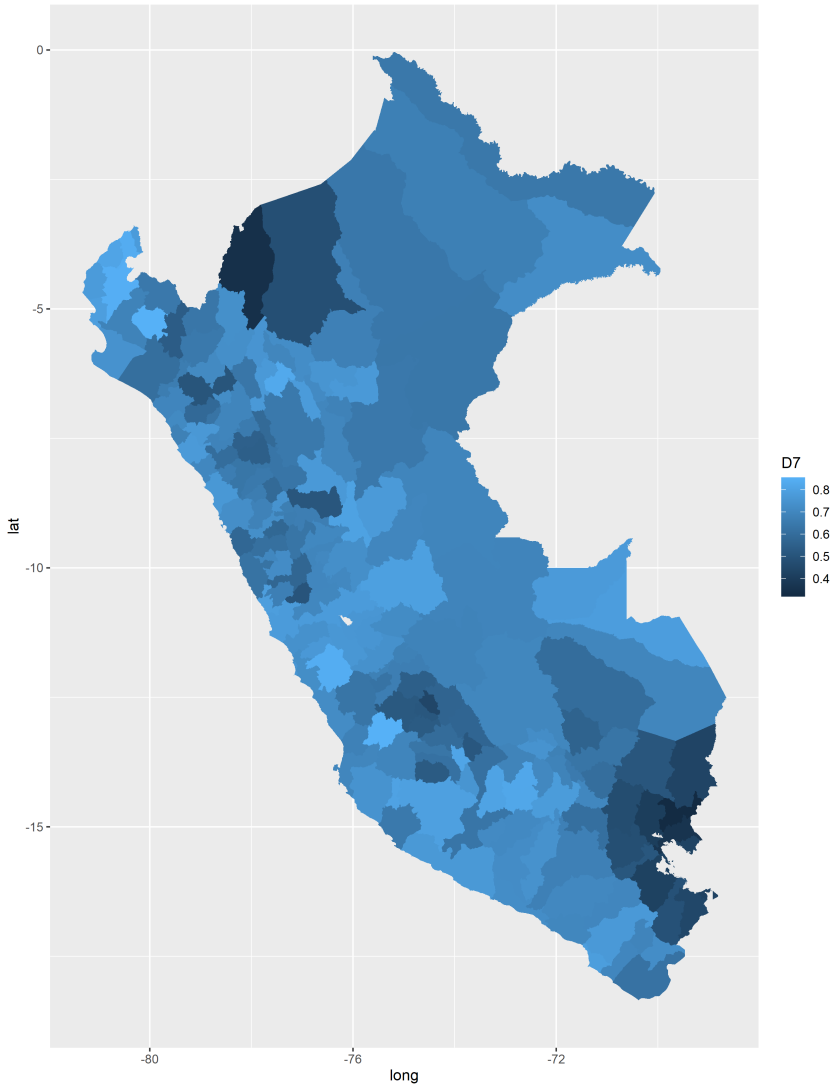
Creando mapas en R con ggplot2

- Se debe combinar estos datos con la variable de interés.
- Se crea el mapa utilizando ggplot2 y se puede personalizar utilizando las distintas herramientas que entrega la librería ggplot2.

Creando mapas en R con ggplot2

```
Mapa <- ggplot() +
    geom_polygon(data=shapes, aes(long, lat, group=group, fill=D7),
                 colour ="black", size = 0.005)
Mapa
```

Creando mapas en R con ggplot2

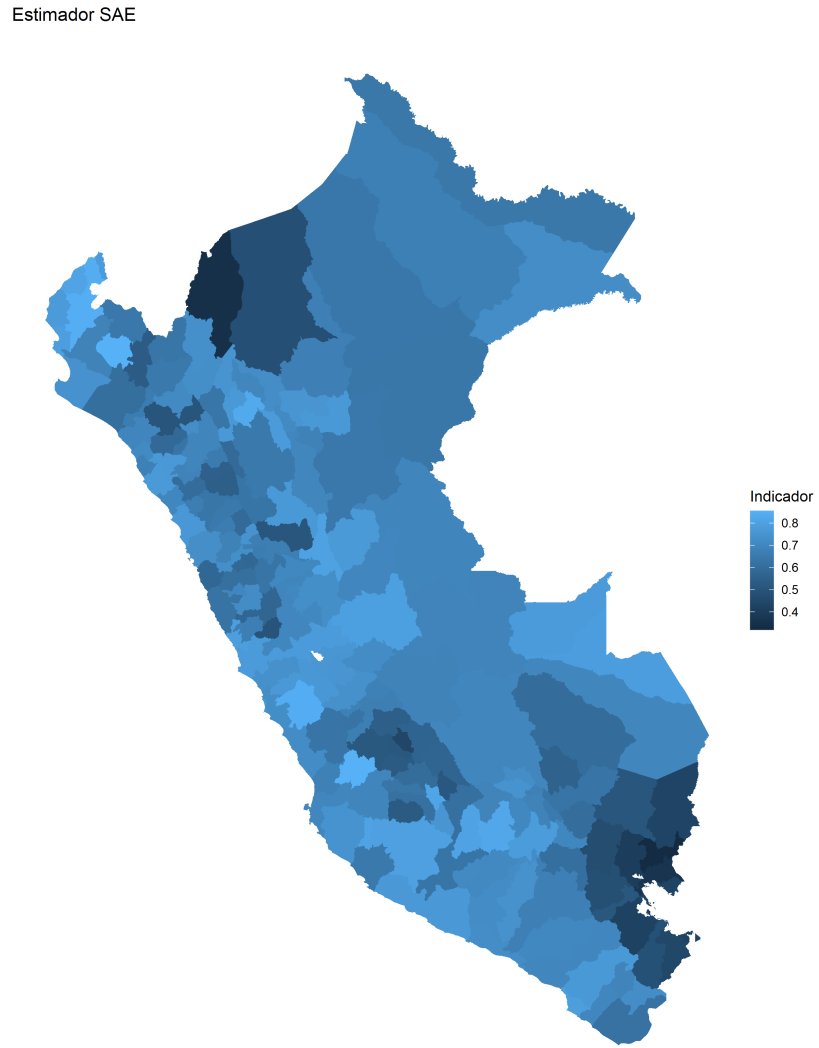


Creando mapas en R con ggplot2

Podemos personalizar el mapa anterior mediante lineas de códigos de ggplot2.

```
Mapa + coord_quickmap() +
  theme_void() +
  labs(fill = "Indicador", title="Estimador SAE")

# ggsave(mapa2, file = "Mapa2.png",
#        width = 8.5, height = 11, type =
# "cairo-png")
```

Creando mapas en R con ggplot2



Discretización de la variable

Es posible discretizar la variable de interés para generar gráficos más limpios con intervalos predeterminados. La librería `RColorBrewer` nos permite acceder a grillas de colores junto a su codificación respectiva.

```
brewer.pal(n = 10, name = 'RdYlGn')
```

```
##  [1] "#A50026" "#D73027" "#F46D43" "#FDAE61" "#FEE08B" "
##  [8] "#66BD63" "#1A9850" "#006837"
```

Discretización de la variable

```
display.brewer.pal(n = 10, name = 'RdYlGn')
```

RdYlGn (divergent)

Discretización de la variable

Los primeros 3 intervalos y el último de la variable discreteada no poseen datos. Utilizamos por tanto, los códigos de colores que corresponden a cada intervalo. Se tienen 10 códigos de colores para los 10 intervalos generados.

```
shapes$discrete_value = cut(100*shapes$D7,
                            breaks=seq(from=0,to=100, length.out=11))

table(shapes$discrete_value)
```

0-10	10-20	20-30	30-40	40-50
0	0	0	14267	55069

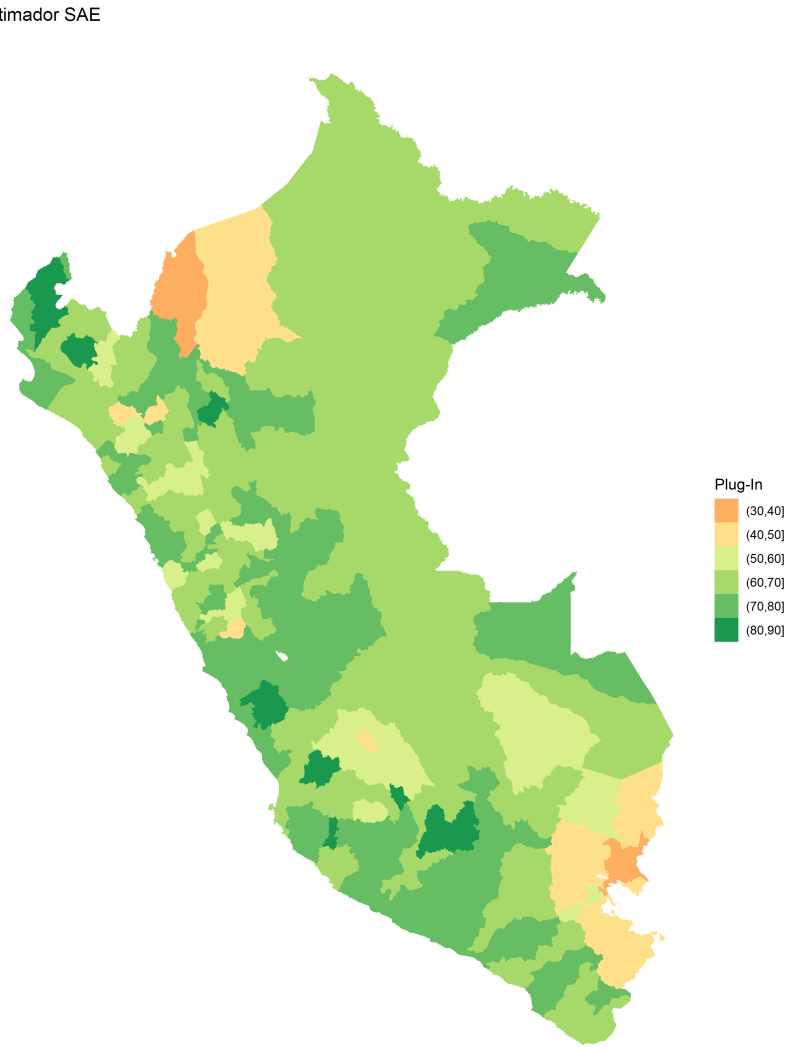
50-60	60-70	70-80	80-90	90-100
82274	276329	236335	30392	0

Discretización de la variable

Generamos el mapa y añadimos los elementos deseados para el mapa final.

```
ggplot() +
  geom_polygon(data=shapes, aes(long, lat, group=group, fill=discrete_value),
               colour ="black", size = 0.005) +
  scale_fill_manual(
    values = c("#FDAE61","#FEE08B","#D9EF8B","#A6D96A","#66BD63","#1A9850"),
    na.value="grey") +
  coord_quickmap()  + theme_void() +labs(fill = "Plug-In", title="Estimador SAE"
```

Discretización de la variable



Departamentos de Colombia

- Es posible obtener la información geoespacial de Colombia en http://www.diva-gis.org/gdata.

Departamentos de Colombia

- Es posible obtener la información geoespacial de Colombia en http://www.diva-gis.org/gdata.
- Colombia se divide en 32 departamentos y un distrito capital Bogotá.

Departamentos de Colombia

- Es posible obtener la información geoespacial de Colombia en http://www.diva-gis.org/gdata.
- Colombia se divide en 32 departamentos y un distrito capital Bogotá.
- El Shape para dicha división viene dado por el archivo `COL_adm1.shp`.

Departamentos de Colombia

- Es posible obtener la información geoespacial de Colombia en http://www.diva-gis.org/gdata.
- Colombia se divide en 32 departamentos y un distrito capital Bogotá.
- El `Shape` para dicha división viene dado por el archivo `COL_adm1.shp`.
- Con el archivo cargado debemos unir la data con la variable de interés, en este caso, utilizaremos números aleatorios

Departamentos en Colombia

```
ohsCol2 <- readOGR("COL_adm/COL_adm1.shp")
ohsColI2 <- fortify(ohsCol2)
grupo2 <- data.frame(id = unique(ohsColI2[ , c("id")]))
grupo2[ , "Porcentaje"] <- runif(nrow(grupo2), 0, 1)
ohsColI2 <- merge(ohsColI2, grupo2, by = "id")
save(ohsColI2, file = "ColData.RData")
```

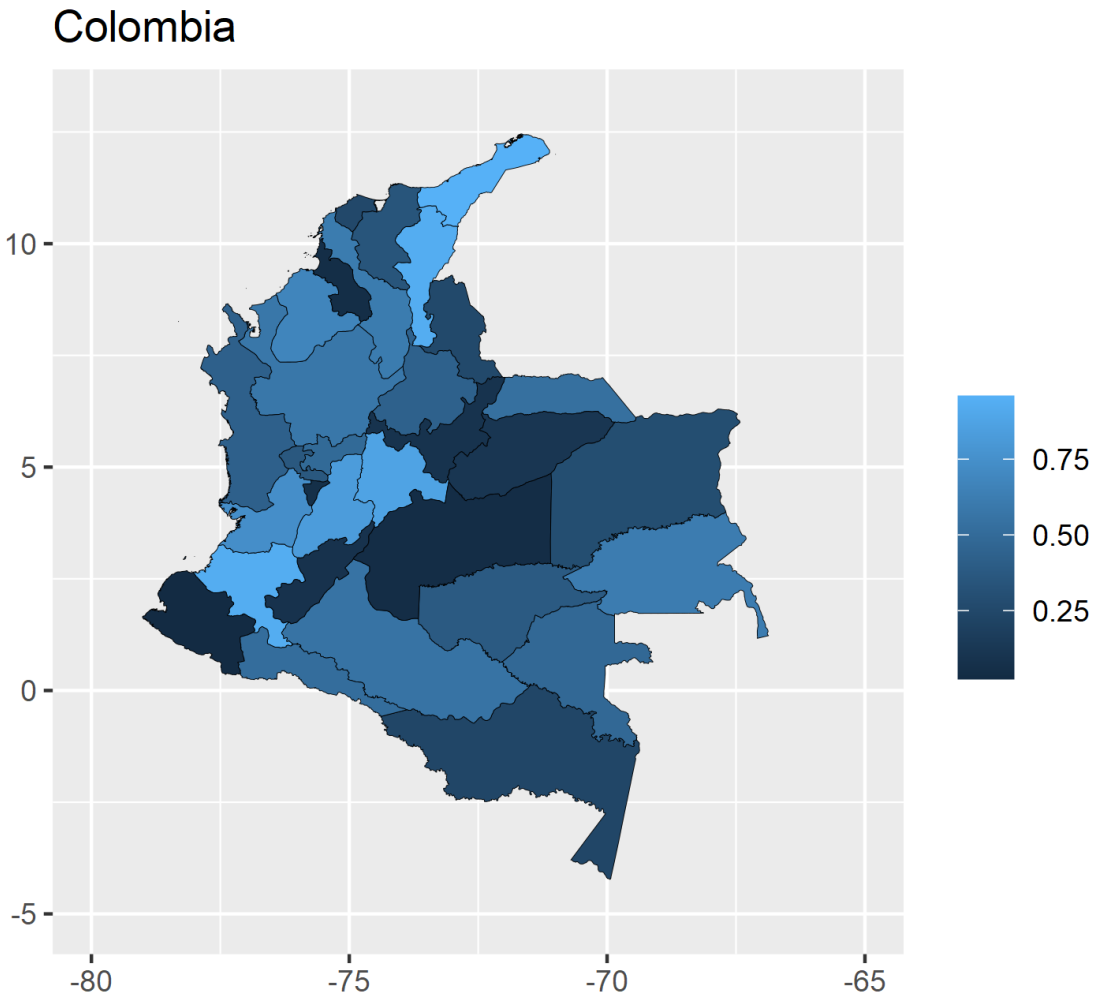
id	long	lat	order	hole	piece	group	Porcentaje
0	-69.43138	-1.078474	1	FALSE	1	0.1	0.2320404
0	-69.42591	-1.096313	2	FALSE	1	0.1	0.2320404
0	-69.42345	-1.104404	3	FALSE	1	0.1	0.2320404
0	-69.41992	-1.111588	4	FALSE	1	0.1	0.2320404
0	-69.41006	-1.131676	5	FALSE	1	0.1	0.2320404
0	-69.39285	-1.148357	6	FALSE	1	0.1	0.2320404

Departamentos de Colombia

Usando la librería ggplot generamos el mapa.

```
mapColDep <- ggplot() +
    geom_polygon(data=ohsColI2, aes(x=long, y=lat, group = group,
                 fill = Porcentaje), colour ="black", size = 0.1) +
    labs(title = "Colombia", fill = "") +
    labs(x="",y="",title="Colombia") +
    scale_x_continuous(limits=c(-80,-65))+
    scale_y_continuous(limits=c(-5,13))
```

Departamentos en Colombia



Departamentos en Colombia

Para guardar el mapa utilizamos la función ggsave.

```
ggsave(mapColDep, file = "mapColDep.png",
       width = 5, height = 4.5, type =
"cairo-png")
```

Mapas con tmap

Mapas en tmap

- La librería tmap funciona de manera similar a ggplot2.

Mapas en tmap

- La librería tmap funciona de manera similar a ggplot2.
- En primer lugar se define el objeto espacial a dibujar utilizando la función `tm_shape()`.

Mapas en tmap

- La librería tmap funciona de manera similar a ggplot2.
- En primer lugar se define el objeto espacial a dibujar utilizando la función `tm_shape()`.
- Para graficar el mapa mediante polígonos utilizando el indicador correspondiente, utilizamos la función `tm_polygons()`.

Mapas en tmap

- La librería tmap funciona de manera similar a ggplot2.
- En primer lugar se define el objeto espacial a dibujar utilizando la función `tm_shape()`.
- Para graficar el mapa mediante polígonos utilizando el indicador correspondiente, utilizamos la función `tm_polygons()`.
- Unimos el indicador a los datos contenidos en `ohsPER2@data` de la información espacial.

Mapas en tmap

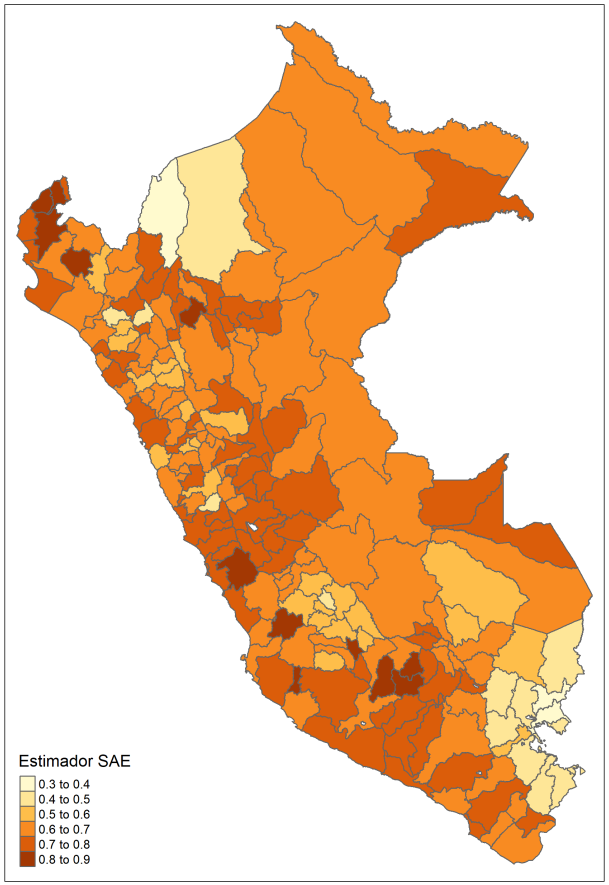
```
library(tmap)
library(maptools)
ohsPER2 <- readOGR("../PER_adm/provincias/PROVINCIAS.shp")
D7 <- readRDS("../D7.rds")

ohsPER2@data <- ohsPER2@data %>% left_join(D7 %>% select(ubigeo,D7),
                                           by = c("IDPROV" = "ubigeo"))
```

Mapas en tmap

Con lo anterior, ya es posible graficar.

```
tm_shape(ohsPER2) + tm_polygons("D7", title = "Estimador SAE")
```

Mapas en tmap



Mapas en tmap

- Utilizando la opción `palette` podemos escoger la paleta de colores para el indicador

Mapas en tmap

- Utilizando la opción `palette` podemos escoger la paleta de colores para el indicador
- anteponiendo un signo - es posible invertir el orden de dicha paleta.

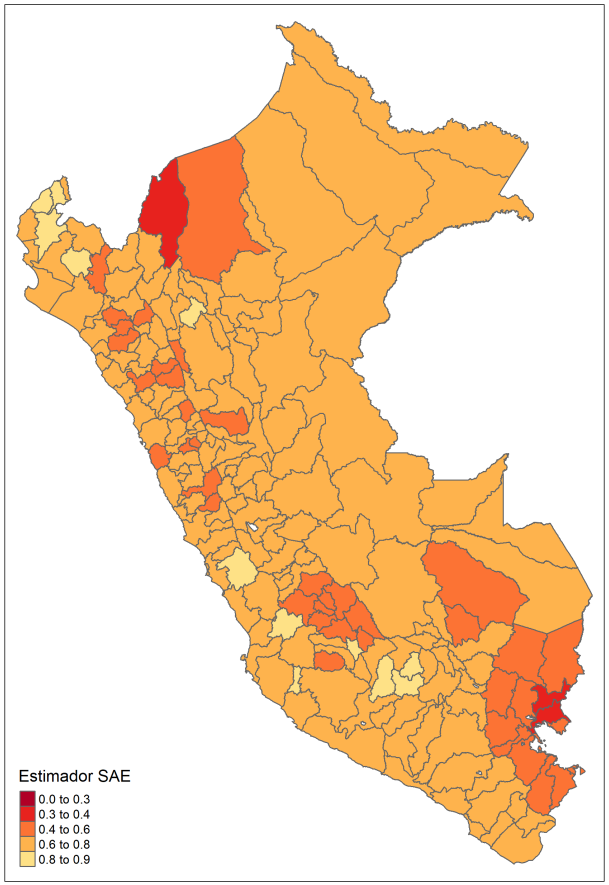
Mapas en tmap

- Utilizando la opción `palette` podemos escoger la paleta de colores para el indicador
- anteponiendo un signo - es posible invertir el orden de dicha paleta.
- Podemos escoger los intervalos utilizando la opción `breaks`.

Mapas en tmap

```
tm_shape(ohsPER2) + tm_polygons("D7", title = "Estimador SAE",
                                palette = "-YlOrRd",
                                breaks = c(0,.4,.5,.6,.7,.8,.9))
```

Mapas en tmap



Mapas en tmap

Para guardar el mapa generado utilizamos la función `tmap_save`

```
tmap_save(mapa, file = "Pics/MapaPeru.png",
          width = 11, height = 8.5, units = "in")
```

Mapa de Chile

- Es posible descargar el mapa vectorial de las comunas de Chile en https://www.bcn.cl/siit/mapas_vectoriales

Mapa de Chile

- Es posible descargar el mapa vectorial de las comunas de Chile en https://www.bcn.cl/siit/mapas_vectoriales
- Utilizando la función `st_read()` leemos la información geoespacial de las 346 comunas de Chile.

Mapa de Chile

```
ChileSP <- st_read("comunas/comunas.shp")
```

```
Simple feature collection with 6 features and 11 fields
geometry type:  MULTIPOLYGON
dimension:      XY
bbox:           xmin: -8133264 ymin: -4748322 xmax: -7828105 ymax: -4017907
projected CRS:  WGS 84 / Pseudo-Mercator
  objectid shape_leng dis_elec cir_sena cod_comuna codregion st_area_sh st_length_
1       48  170038.62       16        8       6204         6  968577420  206184.27
2       29  125730.10       15        8       6102         6  415744636  151911.58
3       30   63026.08       15        8       6103         6  144856484   76355.33
4       31   89840.90       15        8       6104         6  325657168  108874.62
5       78  122626.49       23       11       9121         9  699072708  156680.41
6       79  279936.00       23       11       9103         9 3127304688  360052.12
                                  Region    Comuna     Provincia                       geometry
1 Región del Libertador Bernardo O'Higgins Marchigüe Cardenal Caro MULTIPOLYGON (((-7992819 -4...
2 Región del Libertador Bernardo O'Higgins   Codegua     Cachapoal MULTIPOLYGON (((-7831652 -4...
3 Región del Libertador Bernardo O'Higgins    Coinco     Cachapoal MULTIPOLYGON (((-7892616 -4...
4 Región del Libertador Bernardo O'Higgins  Coltauco     Cachapoal MULTIPOLYGON (((-7906458 -4...
5                  Región de La Araucanía  Cholchol        Cautín MULTIPOLYGON (((-8121756 -4...
6                  Región de La Araucanía     Cunco        Cautín MULTIPOLYGON (((-7992287 -4...
```

Mapa de Chile

- Generamos un indicador aleatorio para cada comuna de Chile o realizar un emparejamiento del indicador sobre las comunas.

```
ChileSP <- ChileSP %>%
  mutate(indicador = runif(346,0,1))
```

Mapa de Chile

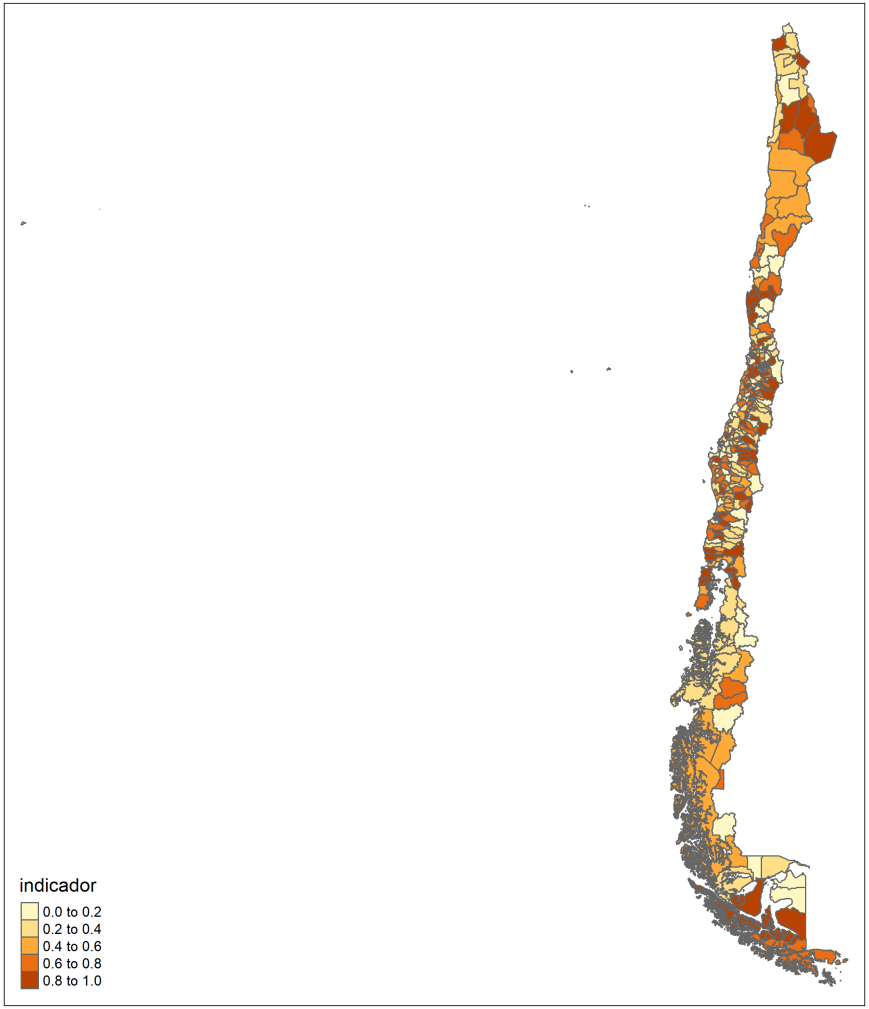
- Generamos un indicador aleatorio para cada comuna de Chile o realizar un emparejamiento del indicador sobre las comunas.
- Es posible graficar el país completo o escoger una Región determinada mediante la función `filter()` de dplyr

```
ChileSP <- ChileSP %>%
  mutate(indicador = runif(346,0,1))
```

Mapa de Chile

```
tm_shape(ChileSP) + tm_polygons("indicador")
```

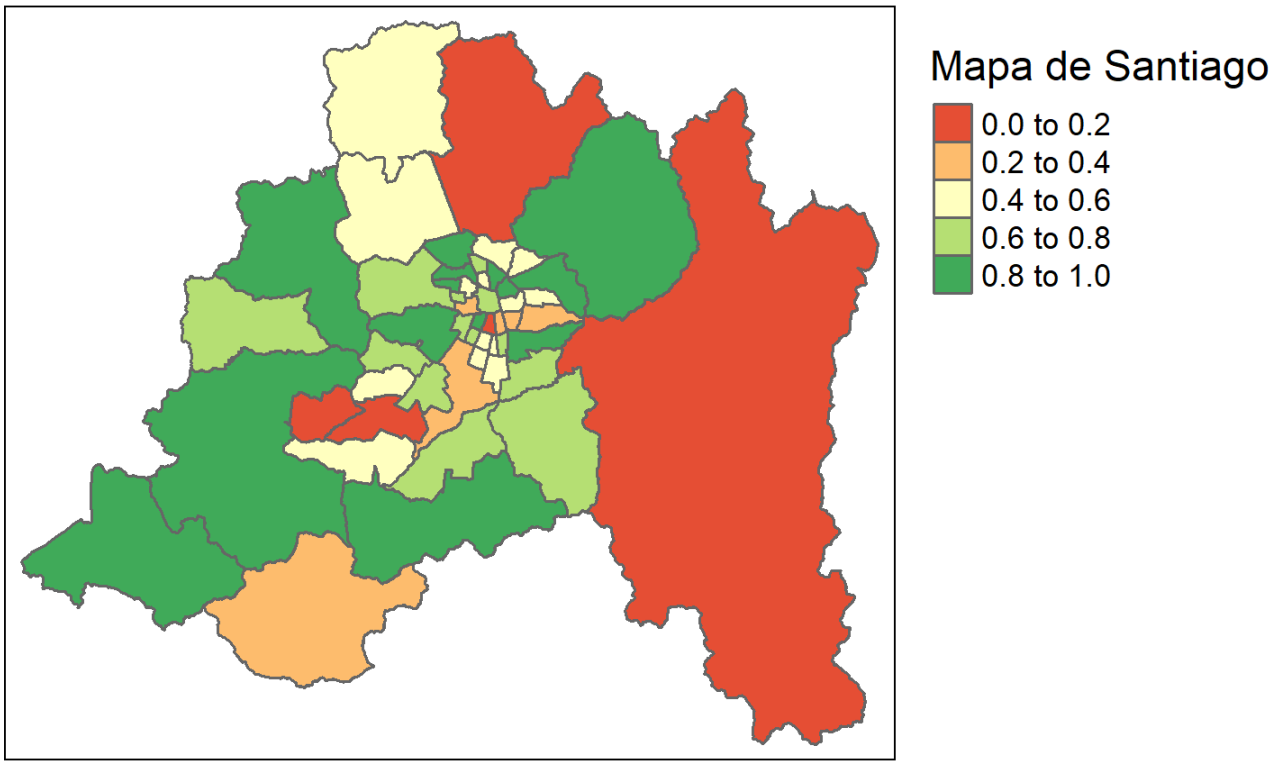
Mapa de Chile



Mapa de Chile: Santiago

```
chile2 <- tm_shape(ChileSP %>% filter(codregion == 13)) +
  tm_polygons("indicador", palette = "RdYlGn",
              title = "Mapa de Santiago") +
  tm_legend(legend.outside = TRUE,
            legend.outside.position="right")
```

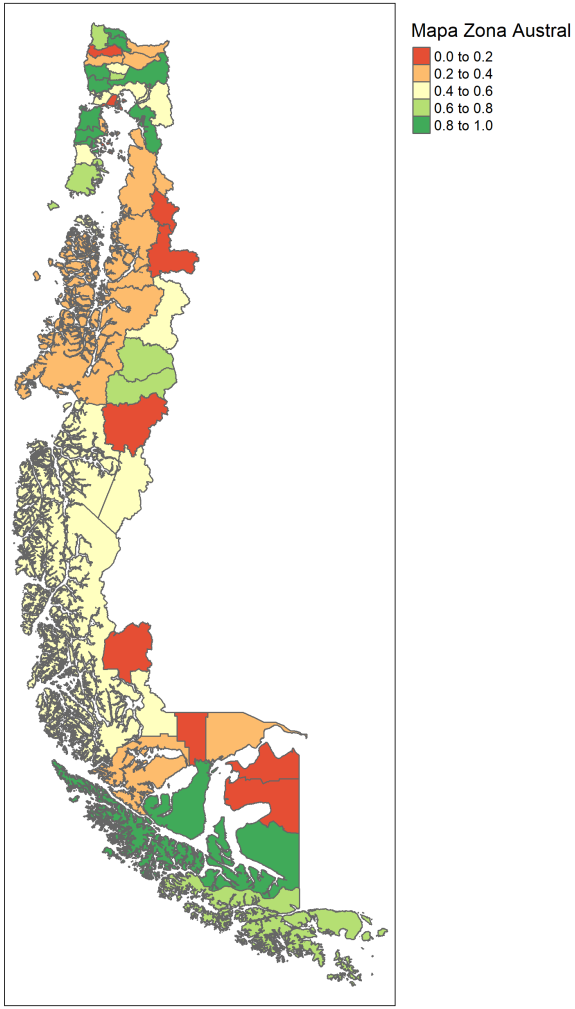
Mapa de Chile: Santiago



Mapa de Chile: Zona Austral

```
tm_shape(ChileSP %>%
           filter(codregion%in% c(10,11,12))) +
  tm_polygons("indicador", palette = "RdYlGn",
              title = "Mapa Zona Austral") +
  tm_legend(legend.outside = TRUE,
            legend.outside.position="right")
```

Mapa de Chile: Zona Austral



Mapas en leaflet

Mapas en leaflet

- Leaflet es una de las librerías de código abierto de Javascript mas utilizada para generar mapas.

Mapas en leaflet

- Leaflet es una de las librerías de código abierto de Javascript mas utilizada para generar mapas.
- A diferencia de lo hecho con ggplot permite generar mapas interactivos a través de populares GIS como CartoDB, OpenStretMaps y Mapboox.

Mapas en leaflet

Las librerías necesarias para la creación de estos mapas se muestra en la siguiente sintaxis:

```
library(dplyr)
library(leaflet)
library(leaflet.extras)
library(leaflet.providers)
library(rgeos)
library(rgdal)
```

Mapa en leaflet

Para empezar a generar mapas interactivos, la función `Leaflet()` genera el entorno para crear el mapa. Podemos generar el mapa mundial con la función `addTiles()`.

```
leaflet() %>% addTiles()
```

Mapa en leaflet

Mapa en leaflet

- Una vez cargada la información geoespacial, se debe indexar el indicador dentro de la data que contiene la lista.

```
ohsPER2 <- readOGR(".../PER_adm/provincias/PROVINCIAS.shp")

ohsPER2@data <- D7 %>% select(ubigeo,D7) %>%
  right_join(ohsPER2@data, by = c("ubigeo"="IDPROV"))
```

Mapa en leaflet

- Una vez cargada la información geoespacial, se debe indexar el indicador dentro de la data que contiene la lista.
- En este caso, las provincias de la información geoespacial están escritas en otro formato, se procede a indexarlas en una nueva `data.frame()` para añadir el indicador a la lista. Esta información se encuentra disponible en `ohsPER@data`.

```
ohsPER2 <- readOGR(".../PER_adm/provincias/PROVINCIAS.shp")

ohsPER2@data <- D7 %>% select(ubigeo,D7) %>%
  right_join(ohsPER2@data, by = c("ubigeo"="IDPROV"))
```

Mapas en leaflet

- Con la información geoespacial actualizada con el indicador, se añade al mapa los polígonos mediante la función `addPolygons()`.

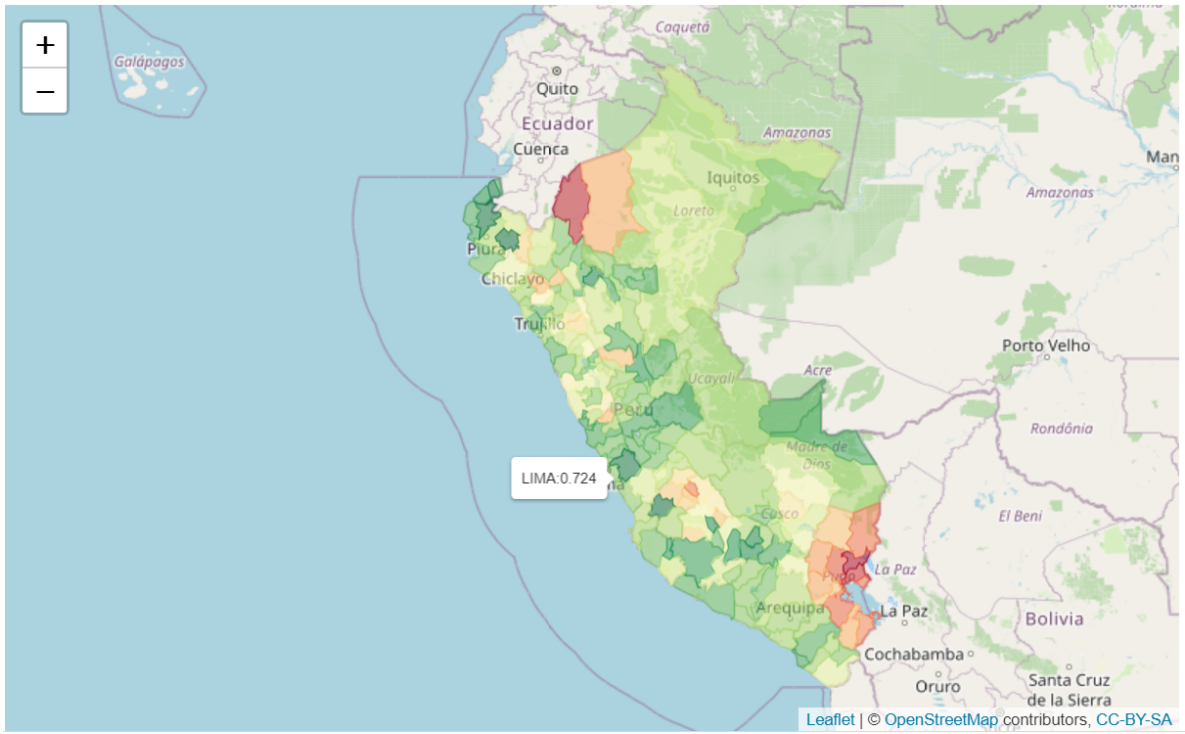
```
d7_pal <- colorNumeric("RdYlGn", domain = ohsPER2@data$D7)

ohsPER2 %>% leaflet() %>% addTiles() %>%
  addPolygons(weight = 1, color = ~d7_pal(D7), fillOpacity = .5,
              label = ~paste0(PROVINCIA,":", round(D7,3)))
```

Mapas en leaflet

- Con la información geoespacial actualizada con el indicador, se añade al mapa los polígonos mediante la función `addPolygons()`.
- Una forma de escoger los colores deseados para mapear el indicador es creando un objeto mediante la función `ColorNumeric()`.

```
d7_pal <- colorNumeric("RdYlGn", domain = ohsPER2@data$D7)

ohsPER2 %>% leaflet() %>% addTiles() %>%
  addPolygons(weight = 1, color = ~d7_pal(D7), fillOpacity = .5,
              label = ~paste0(PROVINCIA,":", round(D7,3)))
```

Mapas en leaflet



Mapas en leaflet

Podemos guardar nuestro mapa en un archivo .html utilizando el código que se muestra a continuación:

```
saveWidget(MAPA,
           file = "PeruTargetGroups.html")
```

¡Gracias!

¡Gracias!