Análisis Exploratorio de Datos Espaciales en R

Rodrigo Tapia McClung

Agosto 10, 2017

En esta práctica vamos a hacer un análisis exploratorio de los datos de las elecciones presidenciales en México a nivel distrital para 1994, 2000, 2006 y 2012. Los datos originales se manipularon para agregar los conteos para los tres principales partidos: PAN, PRI y PRD. En distintos momentos, cada partido ha sido parte de una colición y, para facilitar el manejo de los datos, se usa el nombre corto del partido en lugar de la coalición.

Exploración visual de los datos

Es interesante mapear las variables que queremos analizar. Tenemos 4 momentos de datos de elecciones para los 3 principales partidos y solo toma algo de tiempo hacer que la computadora haga todo el procesamiento de los datos por nosotros. Pero vale la pena ser un poco sensatos en lo que vamos a pedir.

Podemos hacer un mapa de desviaciones estándar que nos muestre la variación espacial alrededor del promedio del porcentaje de votos ganados en cada estado por cada partido en cada elección.

Vamos a ver dos maneras de hacer estos mapas, cada una con sus pros y contras. Primero vamos a trabajar con la librería `GISTools` y después con `ggplot2`.

Requisitos: tener los archivos de trabajo y los de la práctica en alguna carpeta y cambiar a ese directorio de trabajo. Por ejemplo:

```
# Cambiar directorio de trabajo
setwd("C:/Descargas/R/practica1")
```

Asegurarse de tener instaladas las librerías que vamos a usar:

```
# Lista de librerías:
list.of.packages <- c("rgdal", "sp", "GISTools", "RColorBrewer", "ggplot2",
    "reshape2", "grid", "gridExtra")
# Ver qué no está instalado
new.packages <- list.of.packages[!(list.of.packages %in% installed.packages()[,
    "Package"])]
# Si falta algo, instalarlo
if (length(new.packages)) install.packages(new.packages)
```

Abrir SHPs en R

Usaremos `rgdal` para abrir un shapefile en R (`elecciones_simplified.shp`)

```
library(rgdal)
distritos <- readOGR("data", "elecciones_simplified", stringsAsFactors = FALSE,
    GDAL1_integer64_policy = T)
```

```
## OGR data source with driver: ESRI Shapefile
## Source: "data", layer: "elecciones_simplified"
## with 300 features
## It has 25 fields
## Integer64 fields read as doubles:  llavedto POBTOT
```

Si quieres, puedes ver las columnas de la base de datos de este archivo con:

```
colnames(distritos@data)
```

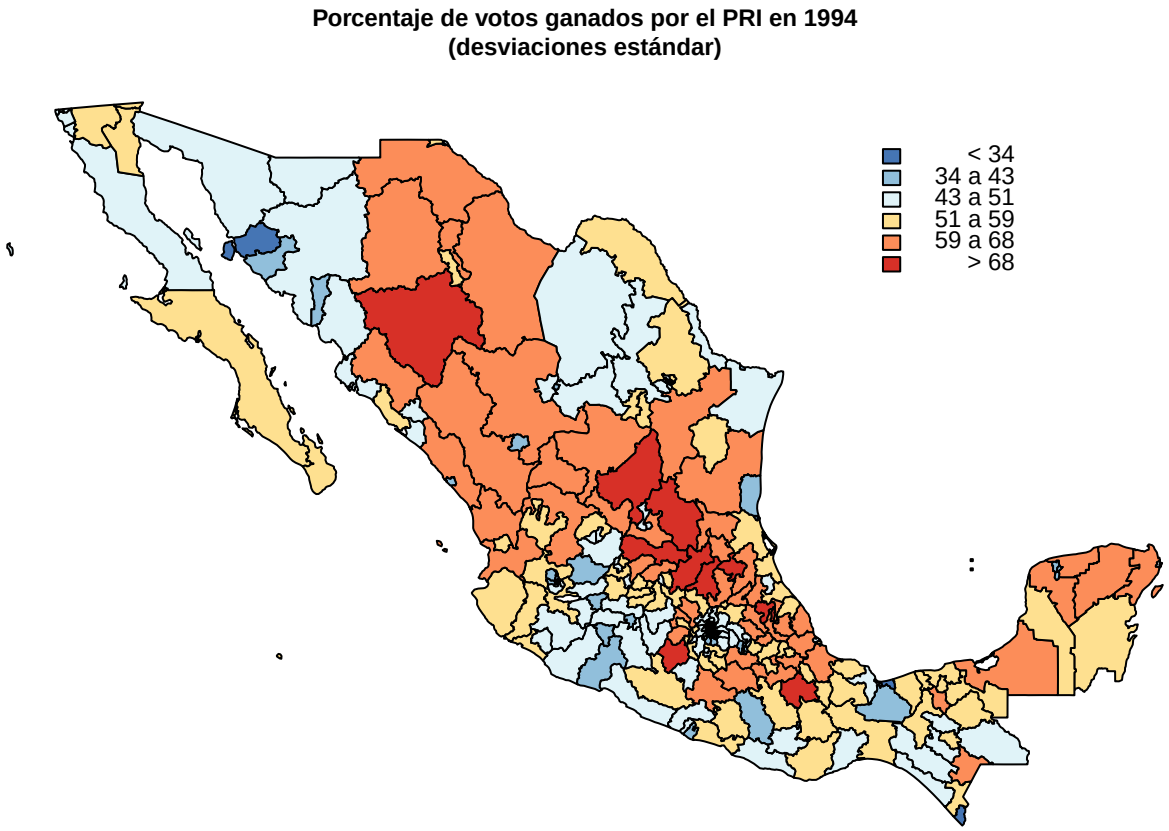
Un mapa de desviaciones estándar - `GISTools`

Cargamos la librería (que asu vez carga una dependencia para usar ciertas rampas de colores):

```
library(GISTools)
#library(RColorBrewer)
```

Definimos nuestro mapa temático:

```
# Definir márgenes para ocupar todo el espacio
par(mar = c(0, 0, 1.5, 0))
# Definir un esquema para colorear el mapa de acuerdo a desviaciones
# estándar
shades <- auto.shading(distritos$PRI94, cutter = sdCuts, n = 6, cols = rev(brewer.pal(6,
    "RdYlBu")))
# Definimos el mapa temático
choropleth(distritos, distritos$PRI94, shades)
# Agregar una leyenda
choro.legend(-95, 32, shades, under = "<", over = ">", between = "a", box.lty = "blank",
    x.intersp = 0.75, y.intersp = 0.75, cex = 0.75)
# Agregar título
title(main = "Porcentaje de votos ganados por el PRI en 1994\n(desviaciones estándar)",
    cex.main = 0.75)
```



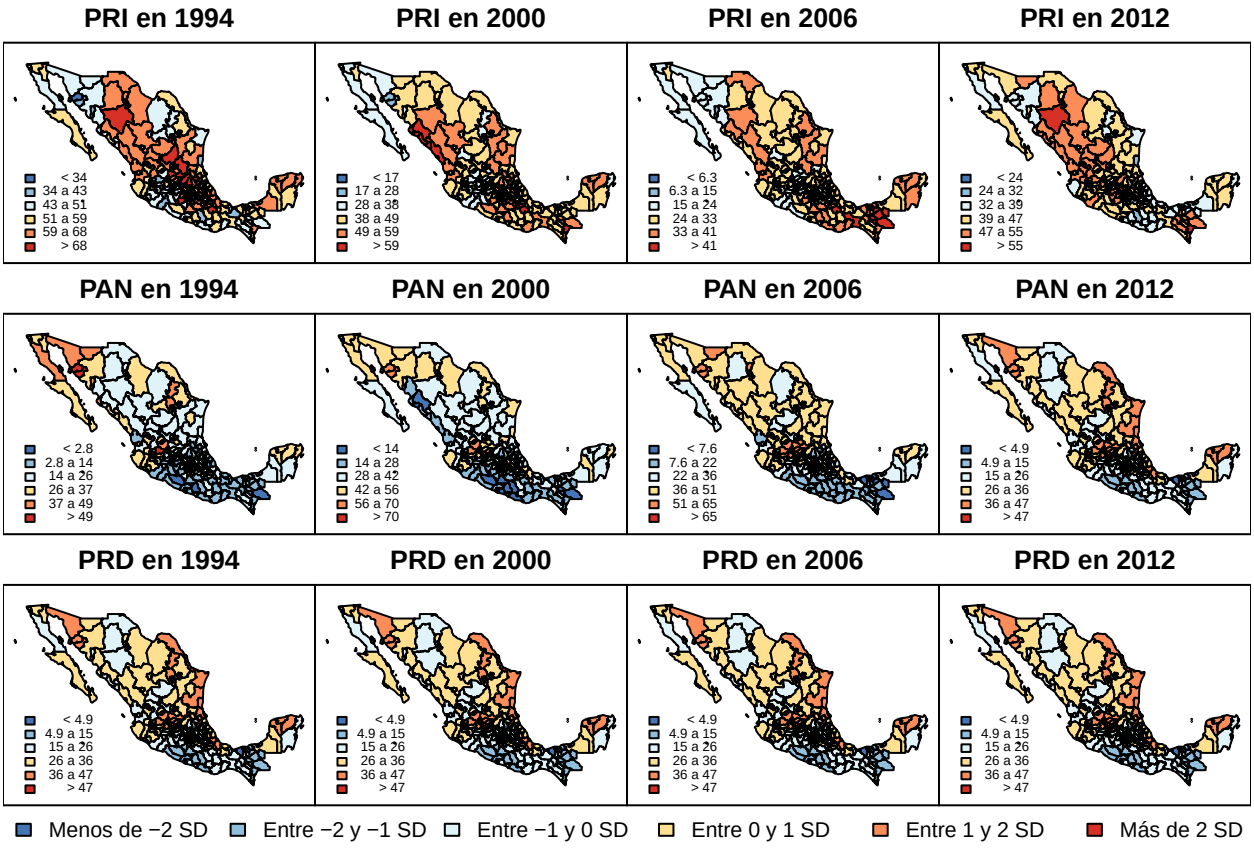
Este mapa muestra que, para 1994, hay muchos distritos con un porcentaje alto de votos ganados por arriba

del promedio. ¿Cómo se compara esto año con año? Vamos a hacer mapas de este estilo para cada año y cada partido y compararlos.

Varios mapas de desviaciones estándar

Podríamos repetir las mismas instrucciones de arriba tantas veces como mapas queramos producir... Pero eso es ineficiente. Mejor vamos a hacer una función:

```
# Definir una función que haga un mapa como el anterior
makeChoro <- function(var, title) {
    shades <- auto.shading(var, cutter = sdCuts, n = 6, cols = rev(brewer.pal(6,
        "RdYlBu")))
    choropleth(distritos, var, shades)
    choro.legend(-118.5, 22.5, shades, under = "<", over = ">", between = "a",
        x.intersp = -1, y.intersp = 0.9, box.lty = "blank", cex = 0.6)
    title(title)
    box(col = "black")
}
# Modificar parámetros del plot para poder acomodar bien: un poco de margen
# abajo para poder acomodar la leyenda, hacer un grid de 3 x 4 y un poco de
# margen arriba entre cada plot para que quepa el título.
op <- par(oma = c(2, 0, 0, 0), mfrow = c(3, 4), mar = c(0, 0, 2, 0))

# Hacer mapa para cada año
pri94 <- makeChoro(distritos$PRI94, "PRI en 1994")
pri00 <- makeChoro(distritos$PRI00, "PRI en 2000")
pri06 <- makeChoro(distritos$PRI06, "PRI en 2006")
pri12 <- makeChoro(distritos$PRI12, "PRI en 2012")
pan94 <- makeChoro(distritos$PAN94, "PAN en 1994")
pan00 <- makeChoro(distritos$PAN00, "PAN en 2000")
pan06 <- makeChoro(distritos$PAN06, "PAN en 2006")
pan12 <- makeChoro(distritos$PAN12, "PAN en 2012")
prd94 <- makeChoro(distritos$PAN12, "PRD en 1994")
prd00 <- makeChoro(distritos$PAN12, "PRD en 2000")
prd06 <- makeChoro(distritos$PAN12, "PRD en 2006")
prd12 <- makeChoro(distritos$PAN12, "PRD en 2012")

legend(-223, 12.5, legend = c("Menos de -2 SD", "Entre -2 y -1 SD", "Entre -1 y 0 SD",
    "Entre 0 y 1 SD", "Entre 1 y 2 SD", "Más de 2 SD"), fill = shades$cols,
    bty = "n", cex = 0.95, y.intersp = 1, x.intersp = 1, horiz = TRUE, xpd = NA)
```



A partir de estos mapas, podemos ver que hay una cierta dinámica en cuanto a la preferencia del electorado tanto en el espacio como en el tiempo. Sin embargo, es un poco difícil comparar tanto los valores reales del porcentaje de votos como de las desviaciones estándar para distintos los años. No obstante, esto nos da una indicación de que hay ciertas zonas del país que se comportan de manera diferente. Esto lo retomaremos más adelante.

Por lo pronto, vamos a hacer otros mapas de desviaciones estándar con otra librería.

Mapas de desviaciones estándar - `ggplot2`

Primero vamos a definir una función para asignarle una etiqueta a cada estado para ver en qué intervalo de desviaciones estándar se encuentra en cada año. Esto lo pudimso hacer con `cutter = sdCuts` en el caso anterior, pero `ggplot2` no implementa esta función, así que tenemos que definirla:

```
sdClass <- function(var){
  mean <- mean(var)
  sd <- sd(var)
  sd.vec <- vector(mode = "character",length = length(var))

  sd.vec[var <= mean-2*sd] <- '<-2' # var <= -2 SD
  sd.vec[var > mean-2*sd & var < mean-sd] <- '-2-1' # -2 SD < var <= -1 SD
  sd.vec[var > mean-sd & var <= mean] <- '-1-0' # -1 SD < var <= 0 SD
  sd.vec[var > mean & var <= mean+sd] <- '0-1' # 0 SD < var <= 1 SD
  sd.vec[var > mean+sd & var < mean+2*sd] <- '1-2' # 1 SD < var < 2 SD
  sd.vec[var >= mean+2*sd] <- '>2' # var => 2 SD
```

```
  # Lo hacemos un factor para que, aunque no haya elementos
  # en alguno de los intervalos, existan y los podamos usar
  sd.vec <- factor(sd.vec, levels = c('<-2', '-2-1', '-1-0', '0-1', '1-2', '>2'))
  return(sd.vec)
}
```

Pero para no confundirnos con los datos que trabajamos anteriormente, volvemos a leer los archivos y los asignamos a nuevas variables:

```
distritos.gg <- readOGR("data", "elecciones_simplified", stringsAsFactors = FALSE,
    GDAL1_integer64_policy = T)
```

```
## OGR data source with driver: ESRI Shapefile
## Source: "data", layer: "elecciones_simplified"
## with 300 features
## It has 25 fields
## Integer64 fields read as doubles:  llavedto POBTOT
```

Ahora creamos un nuevo `data.frame` para esta nueva clasificación, le copiamos los valores de las claves de los distritos y ejecutamos la función que acabamos de definir para cada año:

```
# Definir data frame y copiar datos de clave de entidad
sd.votos <- as.data.frame(distritos.gg[,1])

# Ejecutar la función y clasificar cada año
sd.votos$'sdPRI94' <- sdClass(distritos.gg$PRI94)
sd.votos$'sdPRI00' <- sdClass(distritos.gg$PRI00)
sd.votos$'sdPRI06' <- sdClass(distritos.gg$PRI06)
sd.votos$'sdPRI12' <- sdClass(distritos.gg$PRI12)
sd.votos$'sdPAN94' <- sdClass(distritos.gg$PAN94)
sd.votos$'sdPAN00' <- sdClass(distritos.gg$PAN00)
sd.votos$'sdPAN06' <- sdClass(distritos.gg$PAN06)
sd.votos$'sdPAN12' <- sdClass(distritos.gg$PAN12)
sd.votos$'sdPRD94' <- sdClass(distritos.gg$PRD94)
sd.votos$'sdPRD00' <- sdClass(distritos.gg$PRD00)
sd.votos$'sdPRD06' <- sdClass(distritos.gg$PRD06)
sd.votos$'sdPRD12' <- sdClass(distritos.gg$PRD12)
```

Aquí no se ve ninguna ventaja de `ggplot2` sobre `GISTools` pues estamos repitiendo un pedazo de código tantas veces como mapas queremos tener. Podríamos usar algo como:

```
# Definir un vector de años
anhos <- c('PRI94', 'PRI00', 'PRI06', 'PRI12',
           'PAN94', 'PAN00', 'PAN06', 'PAN12',
           'PRD94', 'PRD00', 'PRD06', 'PRD12')

# Definir data frame y copiar datos de clave de entidad
sd.votos <- as.data.frame(distritos.gg[,1])

# Iterar sobre los años y asignar columnas con nombre sd + elAño
for(a in anhos){
  sd.votos[,paste0("sd",a)] <- do.call("sdClass", list(distritos.gg[[a]]))
}
```

Lo cual actualizará el data frame con 13 columnas: 1 de clave de distrito y 12 de los datos de los años y partidos.

Cargamos la librería:

```
library(ggplot2)
```

Y le pegamos la clasificación que acabamos de hacer a los polígonos de los distritos:

```
# Agregar clases de SD a los distritos
distritos.gg <- merge(distritos.gg, sd.votos, by.x = "CLAVEGEO")
```

Ahora queremos cambiarle el orden a los datos para poder tener una lista *hacia abajo* en vez de *hacia la derecha*. Para esto, usamos el método `melt` de la librería `reshape2`:

```
# Cargar reshape2
library(reshape2)
# Hacer el melt de las clases de SD
sd.votos.melt <- melt(sd.votos, id = c("CLAVEGEO"))
```

Toma un momento para ver la diferencia entre `sd.votos` y `sd.votos.melt`.

Lo malo de usar `ggplot` es que **necesita** usar un *data frame* de `R`. Si te fijas, nuestros distritos son un `SpatialPolygonsDataFrame`. Hay que hacer algo al respecto. Usamos `fortify` para crear un data frame que contenga la geometría de los distritos. Esto hace que perdamos todos los atributos que teníamos, pero si los queremos o necesitamos, se los podemos volver a pegar...

```
# Hacer un data frame de R
distritos_geom.gg <- fortify(distritos.gg, region = "CLAVEGEO")
# Si queremos volver a pegarle los datos originales que ya tenía:
distritos_geom.gg.2 <- merge(distritos_geom.gg, distritos.gg@data, by.x = "id",
    by.y = "CLAVEGEO")
```

Hacer el fortify provoca que cada distrito se separe en segmentos y pierda sus atributos. Al volverle a pegar los datos, cada vértice vuelve a tener sus atributos: el porcentaje de votos y su clasificación de desviaciones estándar por año. Ahora creamos un nuevo data frame en donde le pegamos la clasificación de los intervalos de desviaciones estándar a cada vértice de acuerdo a como está en el *melt* que hicimos:

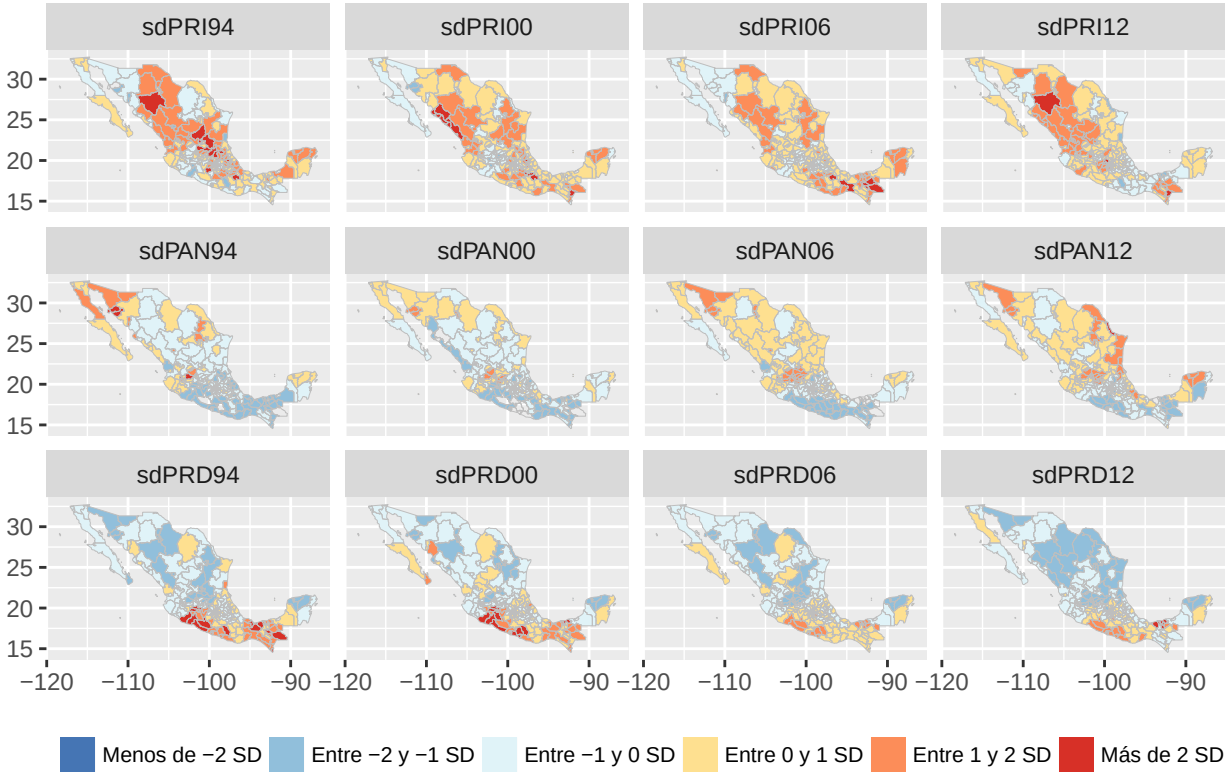
```
# Pegarle los datos de las clasificaciones de SD a cada segmento
plot.data.gg <- merge(distritos_geom.gg, sd.votos.melt, by.x = "id", by.y = "CLAVEGEO")
```

Toda esta vuelta que parece inútil es para poder usar algo super poderoso de `ggplot` que nos permita generar todos los mapas con una sola instrucción:

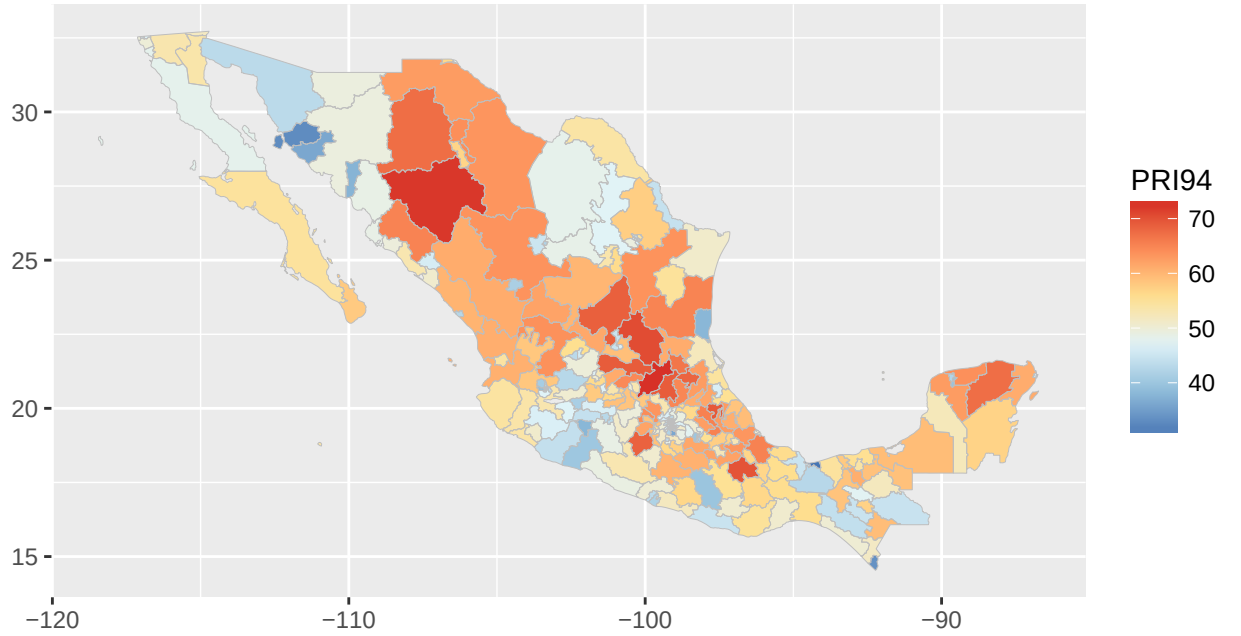
```
# Volvemos a hacer que la clasificacón de las SD sea un factor, porque se
# perdió al hacer el melt...
plot.data.gg$value <- factor(plot.data.gg$value, levels = c('<-2', '-2-1', '-1-0',
  '0-1', '1-2', '>2'))

# Definir el objeto de ggplot
ggplot(data = plot.data.gg, aes(x = long, y = lat, fill = value, group = group)) +
# Agregarle la geometría de los polígonos, colorear los bordes y aspecto 1:1
  geom_polygon() + geom_path(colour = "grey", lwd = 0.1) + coord_equal() +
# Hacer el facet wrap con 4 columnas
  facet_wrap(~variable, ncol = 4) +
# Agregar colores, leyenda y quitar nombres de los ejes
  scale_fill_brewer(palette = "RdYlBu", direction = -1, name = "",
    breaks = c('<-2', '-2-1', '-1-0', '0-1', '1-2', '>2'),
    labels = c('Menos de -2 SD', 'Entre -2 y -1 SD', "Entre -1 y 0 SD", "Entre 0 y 1 SD",
    "Entre 1 y 2 SD", "Más de 2 SD"), drop = FALSE, guide = "legend") +
  labs(x = NULL, y = NULL) +
# Modificar en donde aparece la leyenda
```

```
  theme( legend.position = "bottom",
         legend.key.size = unit(5, "mm"),
         legend.text = element_text(size = 8)) +
  guides(fill = guide_legend(nrow = 1, byrow = TRUE))
```

```
## Warning in grid.Call(L_stringMetric, as.graphicsAnnot(x$label)): font
## metrics unknown for character 0x1f
```

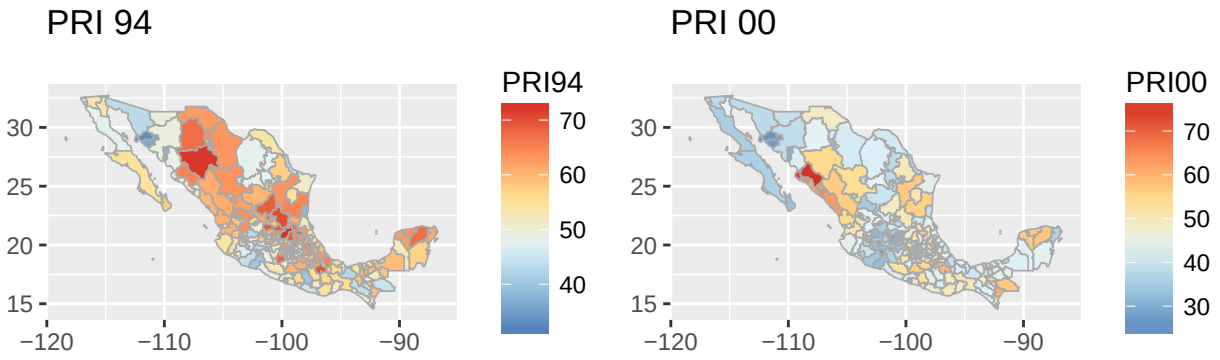


Claro, puedes hacer un solo mapa de desviaciones estándar con `ggplot`. Pero eso se queda de tarea...



Si es muy enredado trabajar los datos para hacer un facet wrap, hay otra forma de hacer los plots en un grid con `ggplot`...

```
# Definir plot para un año
pri94.2 <- ggplot(distritos_geom.gg2, aes(x = long, y = lat, group = group)) +
  geom_polygon(aes(x = long, y = lat, group = group, fill = PRI94), color = "dark grey",
    size = 0.3) + coord_equal() +
  scale_fill_gradientn(colors = rev(brewer.pal(6, 'RdYlBu'))) +
  theme(legend.position = "right") +
  labs(x = NULL, y = NULL, title = "PRI 94", subtitle = "", caption = "")
# Definir plot para otro año
pri00.2 <- ggplot(distritos_geom.gg2, aes(x = long, y = lat, group = group)) +
  geom_polygon(aes(x = long, y = lat, group = group, fill = PRI00), color = "dark grey",
    size = 0.3) + coord_equal() +
  scale_fill_gradientn(colors = rev(brewer.pal(6, 'RdYlBu'))) +
  theme(legend.position = "right") +
  labs(x = NULL, y = NULL, title = "PRI 00", subtitle = "", caption = "")
# Cargar un par de librerías
library(grid)
library(gridExtra)
# Hacer el grid...
grid.arrange(pri94.2, pri00.2, ncol = 2)
```



Un incoveniente es que se repiten las leyendas y `ggplot2` no tiene un método específico para hacer que todas las gráficas compartan la misma leyenda. Se puede hacer pero es un poco enredado... Eso queda para otro taller... Del mismo modo, podrías hacer un *facet wrap* para mostrar la distribución del número total de votos por estado por partido por año.

```
# Regresar a una sola gráfica en toda la página
par(mfrow = c(1,1))
```