Análisis de relaciones de vecindad, dependencia y autocorrelación espacial en R

Rodrigo Tapia McClung

Agosto 11, 2017

Vecindad y dependencia espacial

Queremos estudiar el comportamiento espacial de un conjunto de polígonos. En particular, explorar si presentan autocorrelación espacial global y local. Para ello, recordemos cómo se calcula la I de Moran:

$$I = \frac{n}{\sum_i \sum_{j \neq i} w_{ij}} \frac{\sum_i \sum_{j \neq i} w_{ij}(x_i - \bar{x})(x_j - \bar{x})}{\sum_i (x_i - \bar{x})^2}$$

Y la versión local se puede escribir como:

$$I_i = \frac{(x_i - \bar{x}) \sum_{j \neq i} w_{ij}(x_j - \bar{x})}{\frac{1}{n} \sum_{j \neq i} (x_j - \bar{x})^2}$$

Los valores x son nuestras observaciones y los w_{ij} representan los elementos de una matriz de pesos (renglón i, columna j). Hay que construir esta matriz.

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

```
# Cambiar directorio de trabajo
setwd("C:/Descargas/R/practica3")
```

Abrir SHPs en R

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

```
library(rgdal)
```

```
## Warning: package 'rgdal' was built under R version 3.3.3
```

```
## Loading required package: sp
```

```
## Warning: package 'sp' was built under R version 3.3.3
```

```
## rgdal: version: 1.2-7, (SVN revision 660)
##  Geospatial Data Abstraction Library extensions to R successfully loaded
##  Loaded GDAL runtime: GDAL 2.0.1, released 2015/09/15
##  Path to GDAL shared files: C:/Users/rodrigo.tapia/Documents/R/win-library/3.3/rgdal/gdal
##  Loaded PROJ.4 runtime: Rel. 4.9.2, 08 September 2015, [PJ_VERSION: 492]
##  Path to PROJ.4 shared files: C:/Users/rodrigo.tapia/Documents/R/win-library/3.3/rgdal/proj
##  Linking to sp version: 1.2-4
```

```
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
```

Adyacencia y estadística espacial

Vamos a definir la adyacencia de polígonos. Para ello, primero veremos un ejemplo sencillo al usar puntos y después extenderemos el concepto a polígonos. Después seguiremos con el análisis de la estadística espacial de los porcentajes de elecciones en los estados.

Cargamos la librería `spdep`:

```
library(spdep)
```

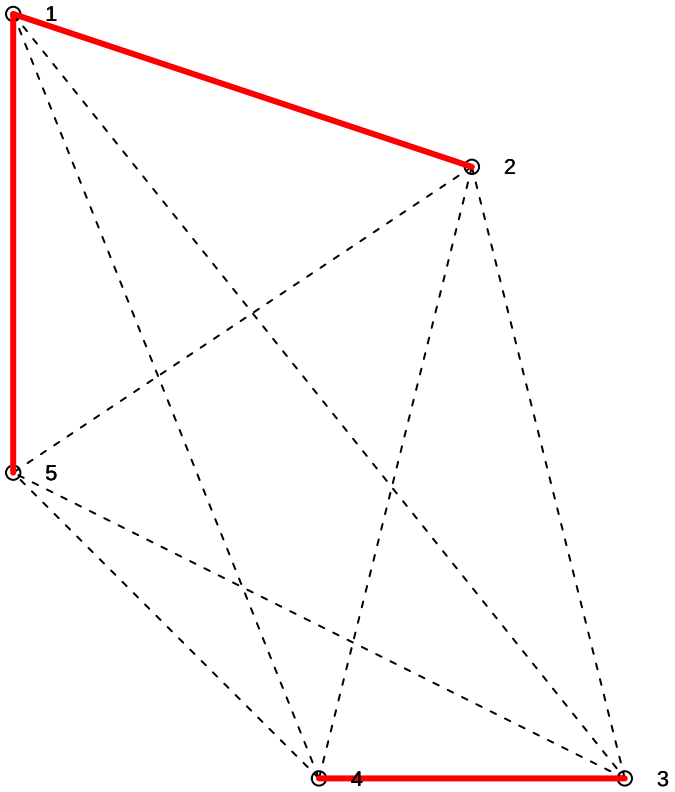
```
## Warning: package 'spdep' was built under R version 3.3.3
```

```
## Loading required package: Matrix
```

Matriz de adyacencia

Una matriz de adyacencia es una manera de expresar si dos entidades son contiguas o no. La manera más fácil de codificar esto es con una matriz de entradas binarias: 0 o 1. Muchas veces estas matrices son simétricas (¡pero no siempre!) pues si A es vecino de B, B suele ser vecino de A. Por ejemplo, considera el siguiente arreglo de puntos:

```
## NULL
```




```
## NULL
```

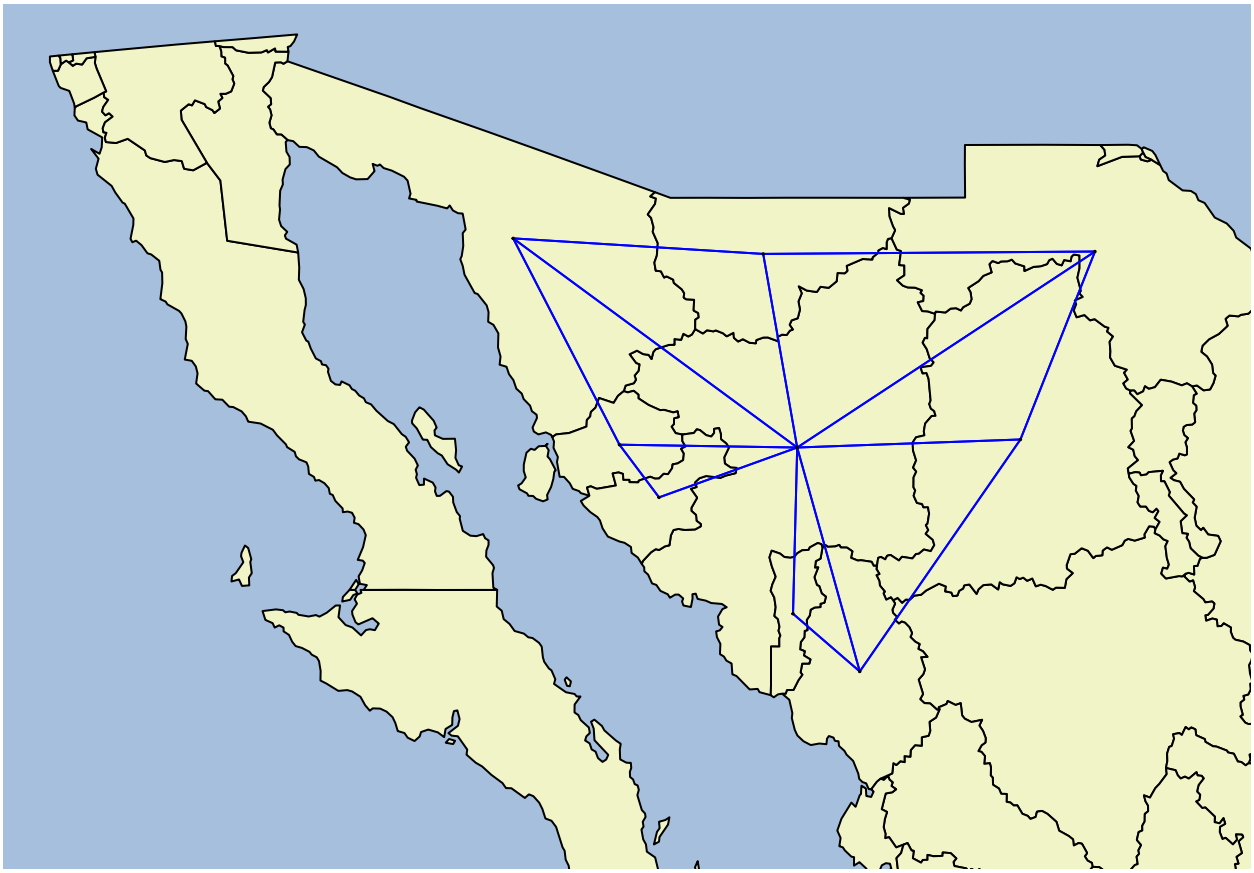
Los puntos 1 y 2 son vecinos, al igual que el 3 y el 4 y el 5 y 1. En forma de matriz esto lo podemos expresar como:

$w_{ij} \rightarrow$	1	2	3	4	5
1	0	**1**	0	0	**1**
2	**1**	0	0	0	0
3	0	0	0	**1**	0
4	0	0	**1**	0	0
5	**1**	0	0	0	0

Los 0s y **1**s son los **pesos**, que en este caso son binarios e indican si hay adyacencia o no. Esta matriz se puede transformar a una que se llama *estandarizada por renglones*. Esto significa sumar los pesos y dividirlos entre el total de la columna. Nuestra matriz anterior estandarizada por renglón se ve así:

$w_{ij} \rightarrow$	1	2	3	4	5
1	0	**0.5**	0	0	**0.5**
2	**1**	0	0	0	0
3	0	0	0	**1**	0
4	0	0	**1**	0	0
5	**1**	0	0	0	0

Si definimos la adyacencia en base a los vecinos de orden mayor, las matrices pueden no ser simétricas. Por ahora, nos vamos a concentrar en adyacencia de primer orden de polígonos. Podemos pensar varias formas en que los polígonos podrían ser vecinos, pero las más comunes son: adyacencia tipo **reina** y tipo **torre**. Por ejemplo, consideremos solo los vecinos del distrito 4 de Sonora:



Aquí podemos ver que **todos** los polígonos que están *pegados* a ese distrito están catalogados como sus vecinos. Ahora tenemos que hacer lo mismo pero para los 300 distritos.

Definir vecinos de contigüidad para los distritos como una lista

```
# Construir la lista de vecinos
distritos.nbq <- poly2nb(distritos, queen = T)  # TRUE: tipo reina
# Convertir la lista de vecinos en una lista de pesos estandarizados por renglón
distritos.nbq.w <- nb2listw(distritos.nbq)
```

Si quieres ver información acerca de la lista de pesos, puedes usar:

```
summary.nb(distritos.nbq)
```

Autocorrelación espacial global y local

Queremos ver si hay alguna indicación de que las observaciones se acumulan en cierta región, o si estamos ante la presencia de un proceso aleatorio. Podemos calcular la `I` de Moran para algún partido y año en particular, digamos el PRD en 1994.

```
# Definimos una variable para no repetir todo el tiempo
var <- distritos$PRD94
moran.test(var, distritos.nbq.w)
```

```
##
```

```
##  Moran I test under randomisation
##
## data:  var
## weights: distritos.nbq.w
##
## Moran I statistic standard deviate = 20.907, p-value < 2.2e-16
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##       0.776560928      -0.003344482       0.001391510
```

```
##          Moran I test under randomisation

## data:  var
## weights: distritos.nbq.w

## Moran I statistic standard deviate = 20.907, p-value < 2.2e-16
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##        0.776560928      -0.003344482       0.001391510
```

Como tiene un valor de `0.77` bastante más alto que el valor esperado, es un indicador de que estamos ante la presencia de autocorrelación espacial global. Ahora calculamos la `I` de Moran local:

```
# Calcular la I de Moran local
lmoran <- localmoran(var, distritos.nbq.w)
```

Puedes ver información acerca de la `I` local con `summary(lmoran)`.

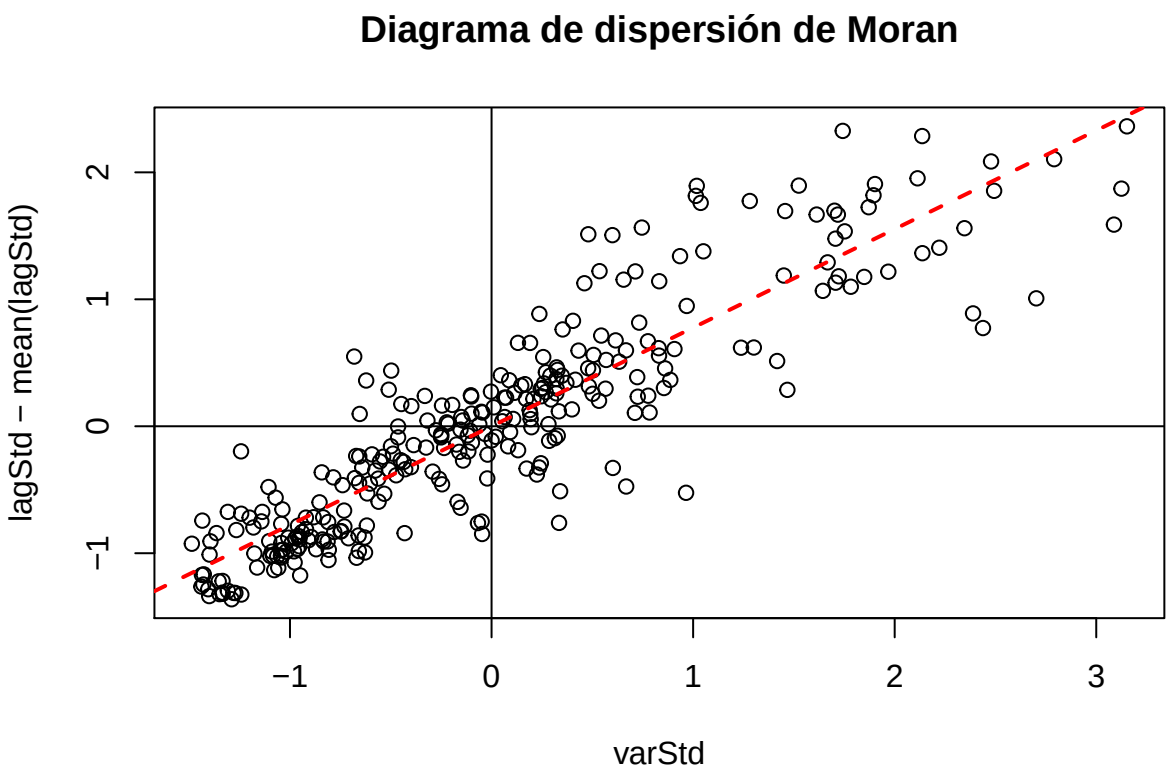
Diagrama de dispersión de Moran

Estandarizamos las variables:

```
lagStd <- lag.listw(distritos.nbq.w, scale(var)) # usamos la variable estandarizada
varStd <- (var - mean(var))/sd(var)
# Es lo mismo que as.vector(scale(var))
```

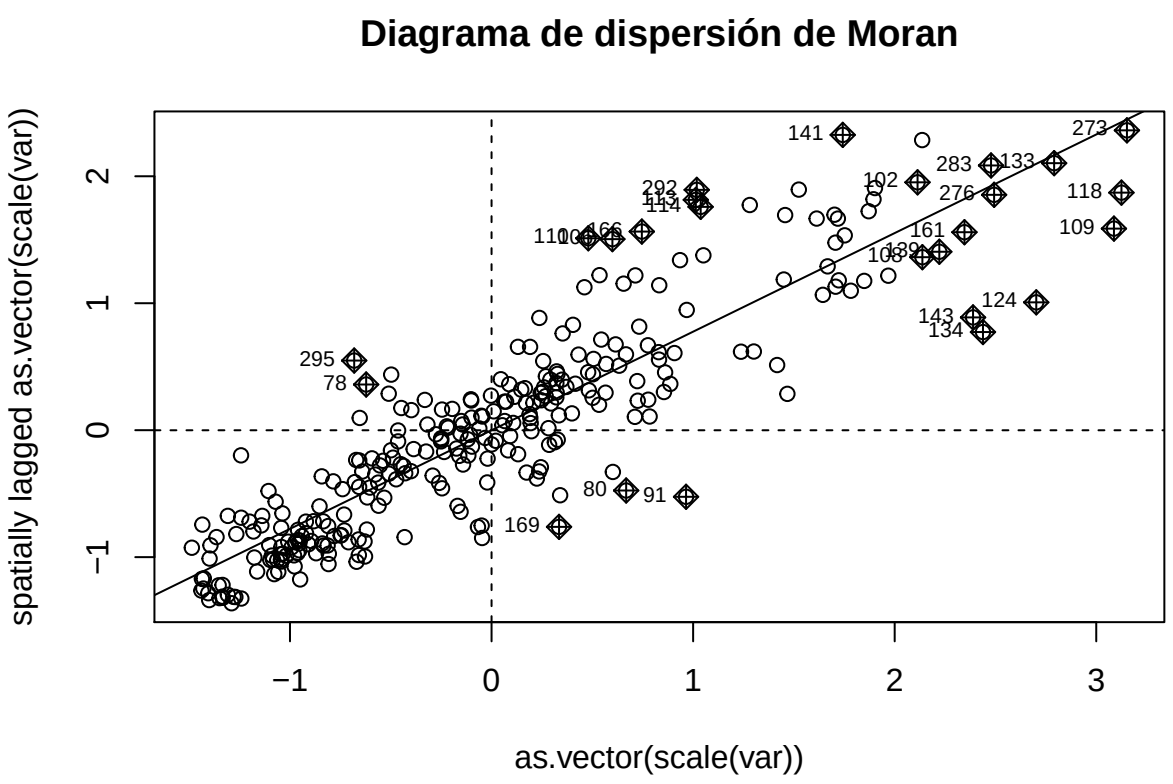
Y hacemos la gráfica:

```
plot(varStd,lagStd-mean(lagStd))
# Ejes que pasan por el origen
abline(h = 0, v = 0)
# Recta de ajuste lineal entre las dos variables
#abline(lm(lagStd ~ varStd), lty = 2, lwd = 2, col = "red")
abline(lm(lagStd-mean(lagStd) ~ varStd), lty = 2, lwd = 2, col = "red")
title("Diagrama de dispersión de Moran")
```



O más rápido, usamos la función `moran.plot()`

```
# Diagrama de dispersión
moran.plot(as.vector(scale(var)), distritos.nbq.w)
title("Diagrama de dispersión de Moran")
```



Vemos que la pendiente es positiva, pues coincide con el valor de la `I` global.

Mapa de cúmulos

Definimos una significancia para nuestros mapas:

```
# Definir significancia
significancia <- 0.05
```

y un vector para almacenar el cuadrante en el que se encuentra cada punto en el diagrama de Moran:

```
# Definir vector de cuadrante del plot de Moran
cuadrante <- vector(mode = "numeric",length = length(var))
```

En el diagrama de dispersión de Moran, el eje `x` es la variable `varStd` y el eje `y` es `lagStd`. Así que encontramos en cuál cuadrante está cada punto:

```
# Definición de cúmulos
cuadrante[varStd > 0 & lagStd > 0] <- 1 # High-High
cuadrante[varStd < 0 & lagStd < 0] <- 2 # Low-Low
cuadrante[varStd < 0 & lagStd > 0] <- 3 # Low-High
cuadrante[varStd > 0 & lagStd < 0] <- 4 # High-Low
cuadrante[lmoran[,5] > significancia] <- 0 # Not significant
```

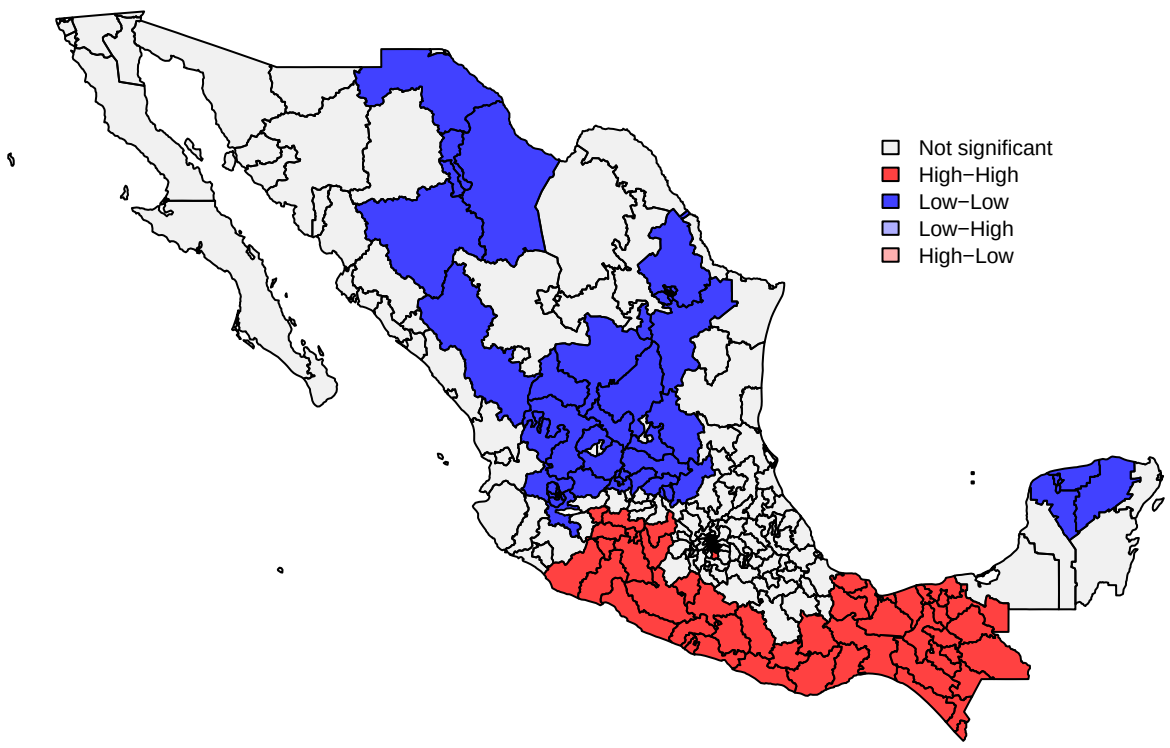
Y definimos los colores que vamos a usar:

```
# Colores para los cúmulos
cColors <- c(rgb(0.74, 0.74, 0.74, alpha = 0.2), rgb(1, 0, 0, alpha = 0.75),
```

```
    rgb(0, 0, 1, alpha = 0.75), rgb(0.58, 0.58, 1, alpha = 0.75), rgb(1, 0.58,
        0.58, alpha = 0.75))
# gris, rojo, azul, azul claro y rojo claro not significant, high-high,
# low-low, low-high, high-low
```

El mapa de cúmulos:

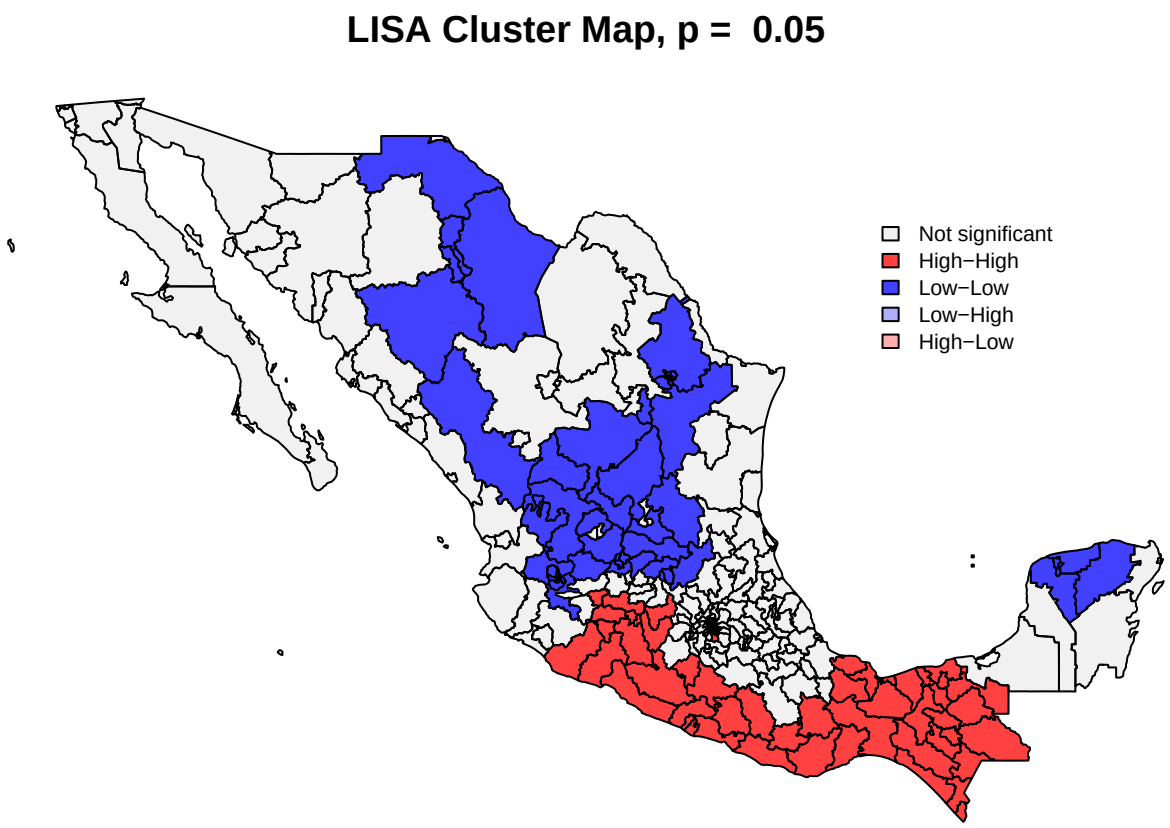
```
# Definir márgenes para ocupar todo el espacio
par(mar = c(0, 0, 1, 0))
# Primer mapa plot not significant
plot(distritos[cuadrante == 0, ], col = cColors[1], pch = 16, cex = 0.75)
# plot high-highs
plot(distritos[cuadrante == 1, ], col = cColors[2], add = T, pch = 16, cex = 0.75)
# plot low-lows
plot(distritos[cuadrante == 2, ], col = cColors[3], add = T, pch = 16, cex = 0.75)
# plot low-highs
plot(distritos[cuadrante == 3, ], col = cColors[4], add = T, pch = 16, cex = 0.75)
# plot high-lows
plot(distritos[cuadrante == 4, ], col = cColors[5], add = T, pch = 16, cex = 0.75)
legend(-95, 30, legend = c("Not significant", "High-High", "Low-Low", "Low-High",
    "High-Low"), fill = cColors, bty = "n", cex = 0.7, y.intersp = 1, x.intersp = 1)
title(paste("LISA Cluster Map, p = ", significancia))
```



O de una forma más rápida, pintamos todos con una sola instrucción:

```
# Definir márgenes para ocupar todo el espacio
par(mar = c(0, 0, 1, 0))
```

```
intervalos <- c(0, 1, 2, 3, 4)
plot(distritos, col = cColors[findInterval(cuadrante, intervalos)])
legend(-95, 30, legend = c("Not significant", "High-High", "Low-Low", "Low-High",
    "High-Low"), fill = cColors, bty = "n", cex = 0.7, y.intersp = 1, x.intersp = 1)
title(paste("LISA Cluster Map, p = ", significancia))
```



Pseudo-significancia y permutaciones

Veamos qué tan confiable es el resultado de la `I` local que obtuvimos. Para esto, podemos hacer permutaciones de las cuentas observadas en cada estado para ver qué tan diferente es de una distribución normal, pero manteniendo las ubicaciones espaciales. Definimos un número de permutaciones a realizar:

```
# Definir el número de simulaciones
sims <- 999
```

Definimos una matriz y un vector para guardar los resultados de las simulaciones.

```
# Definir una matriz y un vector para guardar dos resultados
# La I local simulada
sim.I <- matrix(0, sims, 300)
# Los p-values simulados
sim.p <- matrix(0, 1, 300)
```

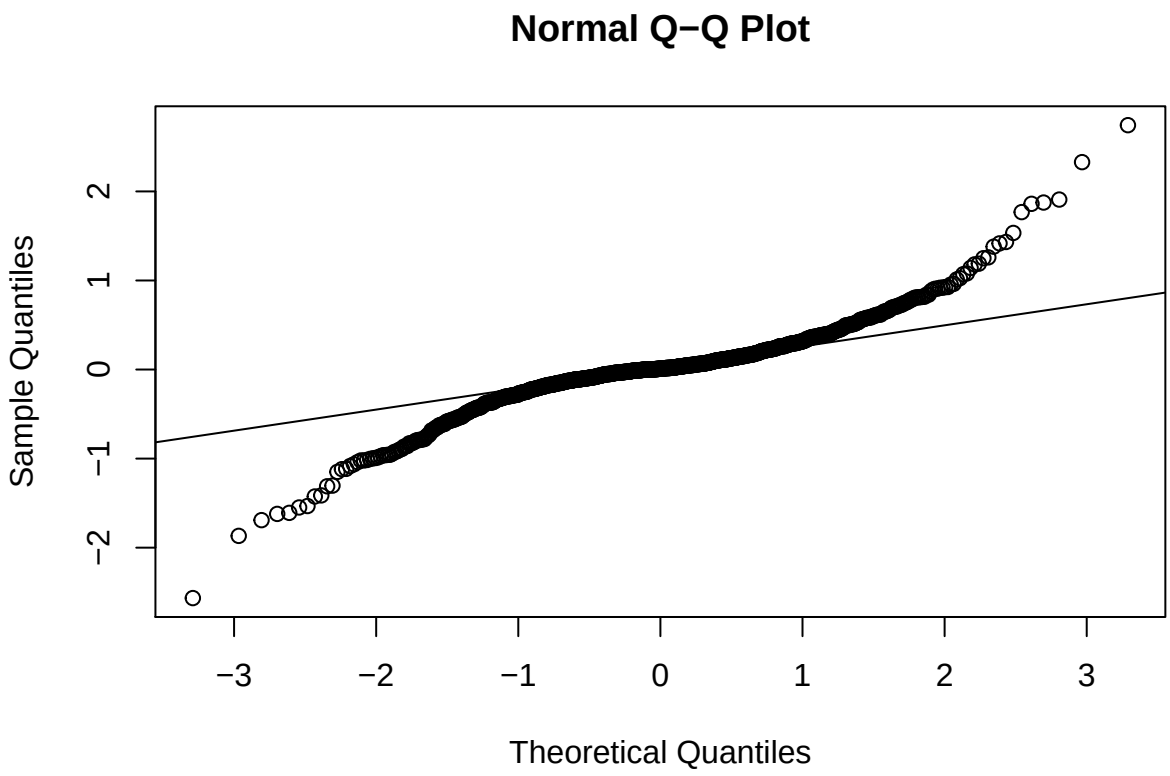
Y encontramos los valores de la `I` local para cada simulación:

```
# Encontrar la I de Moran local para cada simulación
for(i in 1:sims){
```

```
  sim.I[i,] <- localmoran(sample(var), distritos.nbq.w)[,1]
}
```

Podemos hacer una gráfica QQ para ver si nuestras simulaciones se parecen, o no, a una distribución normal:

```
# QQ plots for polygon
qqnorm(sim.I[,1])
qqline(sim.I[,1])
```



Queremos comparar `M` simulaciones de los datos conel valor observado y contar cuántas son mayores o menores que el valor local observado. La pseudo-significancia la podemos definir como

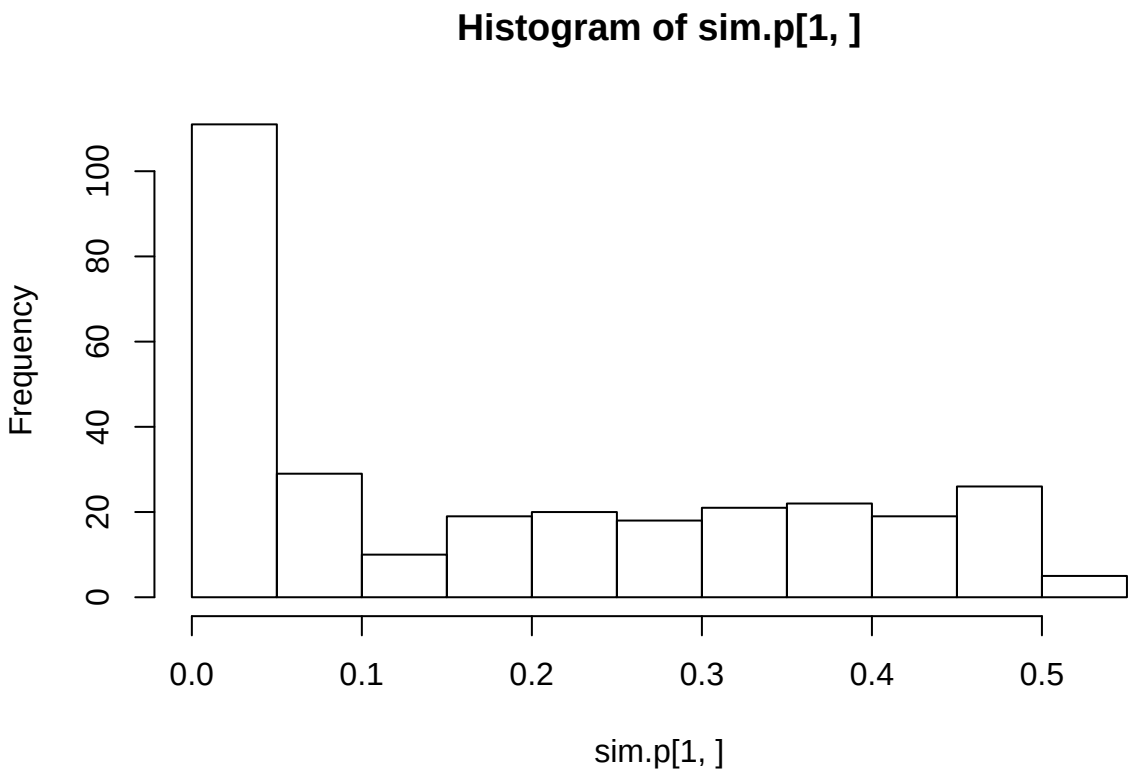
$$p = \frac{M+1}{R+1}$$

donde `M` es el número de veces que la estadística simulada es mayor (menor) o igual que la observada y `R` es el número de simulaciones.

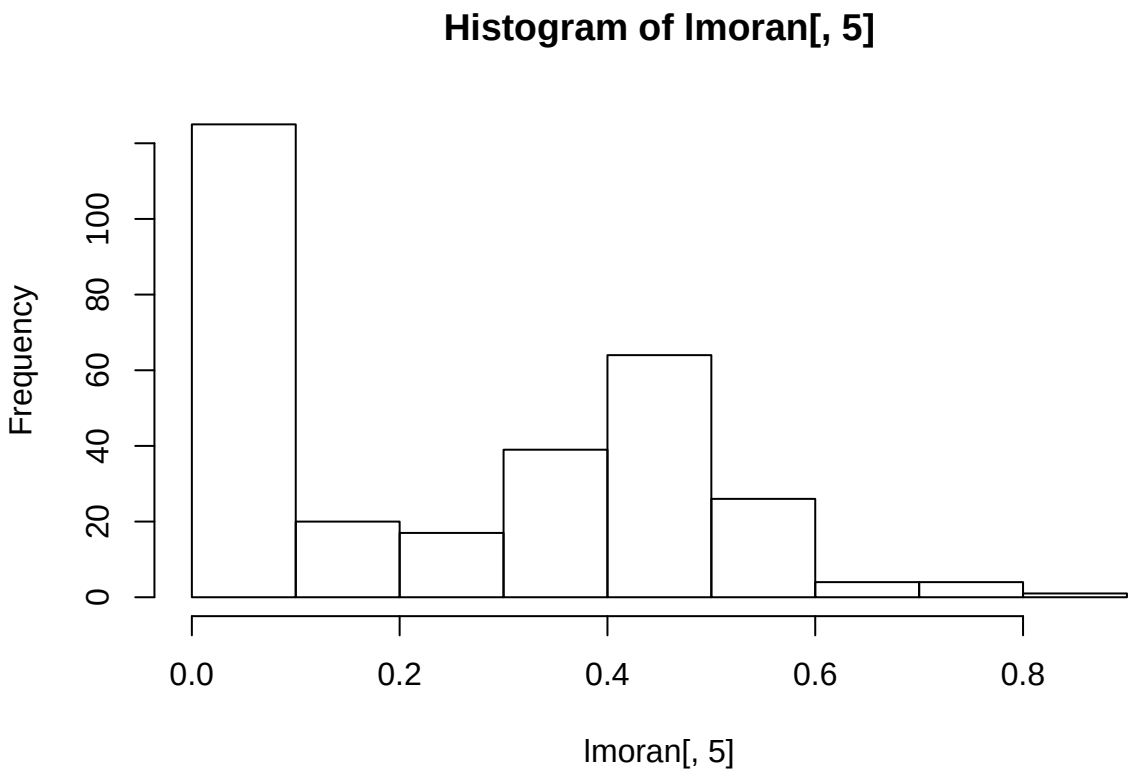
```
# Calcular el valor p definido arriba
for(i in 1:300){
  ifelse(lmoran[i,1]>0, larger <- sim.I[,i]>=lmoran[i,1], larger <- sim.I[,i]<=lmoran[i,1])
  sim.p[i] <- (sum(larger == T)+1)/(sims+1)
}
```

Con esto, podemos comparar la distribución de las significancias para la `I` de Moran observada y las simuladas:

```
# Comparar los histogramas de los p-values
hist(sim.p[1,])
```




```
hist(lmoran[,5])
```



Y por último, podemos hacer un mapa de cúmulos tomando en cuenta estas pseudo-significancias. Como antes, definimos un vector para los cuadrantes de los valores simulados:

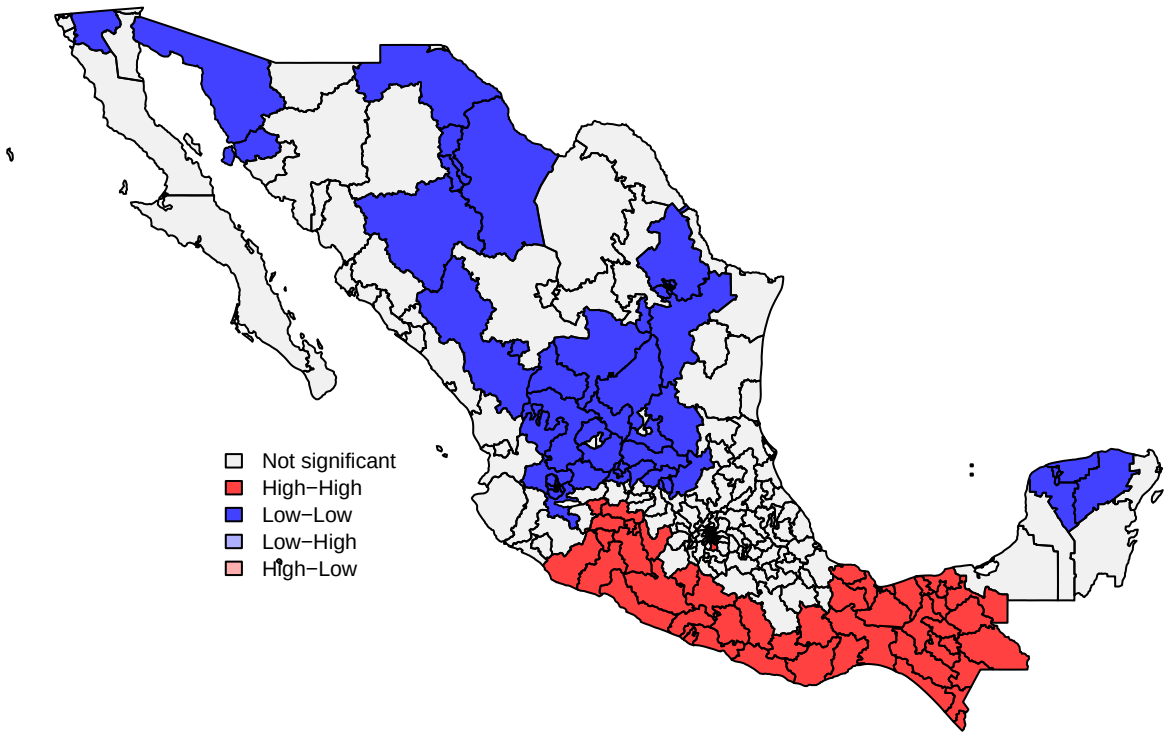
```
# Definir vector de cuadrante del plot de Moran de las simulaciones
cuadrante.sim <- vector(mode = "numeric", length = length(var))
```

Encontramos en cuál cuadrante está cada punto:

```
# Definición de cúmulos
cuadrante.sim[varStd > 0 & lagStd > 0] <- 1 # High-High
cuadrante.sim[varStd < 0 & lagStd < 0] <- 2 # Low-Low
cuadrante.sim[varStd < 0 & lagStd > 0] <- 3 # Low-High
cuadrante.sim[varStd > 0 & lagStd < 0] <- 4 # High-Low
cuadrante.sim[sim.p > significancia] <- 0 # Not significant
```

Y hacemos el mapa de las simulaciones:

```
# Definir márgenes para ocupar todo el espacio
par(mar = c(0, 0, 1, 0))
plot(distritos, col = cColors[findInterval(cuadrante.sim, intervalos)])
legend(-113, 22, legend = c("Not significant", "High-High", "Low-Low", "Low-High",
    "High-Low"), fill = cColors, bty = "n", cex = 0.7, y.intersp = 1, x.intersp = 1)
title(paste("LISA Cluster Map, p = ", significancia, " - ", sims, " simulaciones"))
```

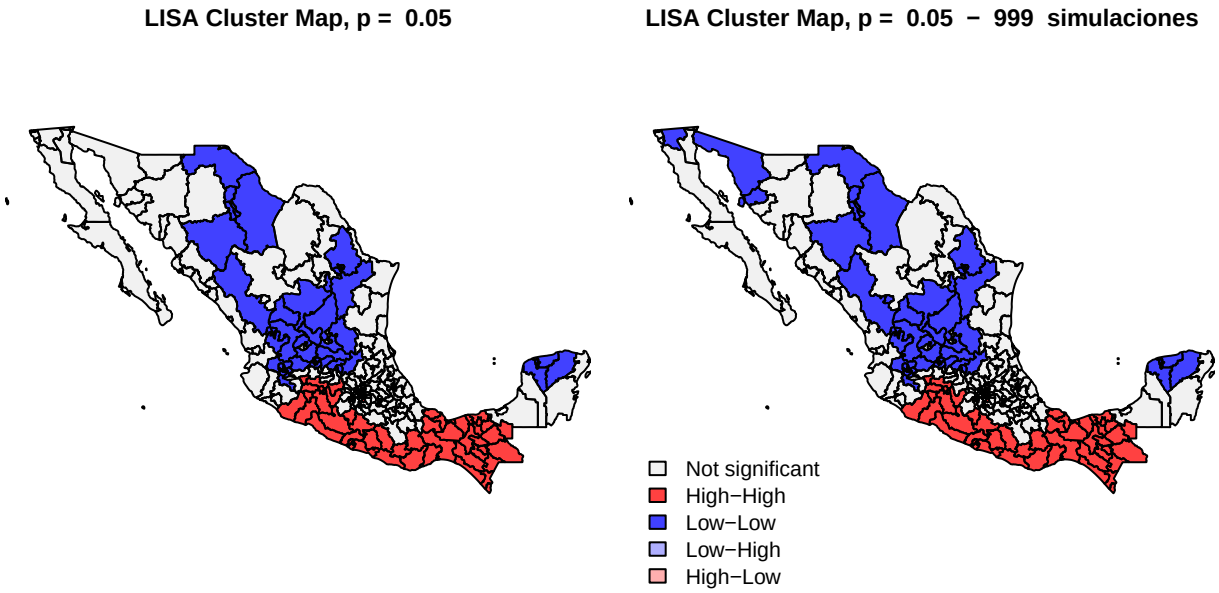


Para comparar los dos mapas de cúmulos, los podemos poner juntos:

```
# Definir que tendremos un gráfica de 1 renglón y dos columnas
par(mfrow = c(1, 2))
# Sin margen arriba, un poco de margen entre cada plot y definir tipo de
# región cuadrada
op <- par(oma = c(0, 0, 0, 0), mar = c(0, 1, 0, 0), pty = "s")

# Primer mapa
plot(distritos, col = cColors[findInterval(cuadrante, intervalos)], xaxs = "i",
    yaxs = "i")
title(paste("LISA Cluster Map, p = ", significancia), cex.main = 0.75)

# Segundo mapa
plot(distritos, col = cColors[findInterval(cuadrante.sim, intervalos)], xaxs = "i",
    yaxs = "i")
legend("bottomleft", legend = c("Not significant", "High-High", "Low-Low", "Low-High",
    "High-Low"), fill = cColors, bty = "n", cex = 0.7, y.intersp = 1, x.intersp = 1)
title(paste("LISA Cluster Map, p = ", significancia, " - ", sims, " simulaciones"),
    cex.main = 0.75)
```




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