GEOG 413 Unit 4

Aaron Goodman

2025-03-02

Libraries

```
library(tmap)
library(sf)
```

```
## Linking to GEOS 3.12.2, GDAL 3.9.3, PROJ 9.4.1; sf_use_s2() is TRUE
```

```
library(sp)
```

```
## Warning: package 'sp' was built under R version 4.4.3
```

```
library(spdep)
```

```
## Warning: package 'spdep' was built under R version 4.4.3

## Loading required package: spData

## Warning: package 'spData' was built under R version 4.4.3

## To access larger datasets in this package, install the spDataLarge
## package with: `install.packages('spDataLarge',
## repos='https://nowosad.github.io/drat/', type='source')`
```

[1]. Find a polygon shapefile with at least 50 polygons that contains at least one interesting variable that you will examine in questions 2 and 3. (Do not use data from one of the Module 4 exercises.)

```
acspotus <- st_read("../data/cty_acs_potus_rf_shp/cty_acs_potus_rf.shp")
```

```
## Reading layer `cty_acs_potus_rf' from data source
##   `F:\MAGIST\23_25w_GEOG_413_AppliedGeospatialStats\U4_Spatial-Autocorrelation\data\cty_acs_potus_rf
##   using driver `ESRI Shapefile'
## Simple feature collection with 3089 features and 59 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -124.849 ymin: 24.39631 xmax: -66.88544 ymax: 49.38448
## Geodetic CRS:  NAD83
```

```
capotus <- acspotus[acspotus$state %in% "CA",]
```

[a]. Read the shapefile into an object in R – you might have to make sure that it is a spatial object.

```
capotus2 <- as(capotus, "Spatial")
```

[b]. Use qtm to generate a plot of your study region.

```
qtm(capotus)
```

[c]. Use the functions within spdep to generate queen and rook contiguity neighbors (order 1) for your study region. Provide summary statistics for your rook and queen neighbor objects and plot your two neighbor maps.

```
# Queen contiguity
capotuswm_q <- poly2nb(capotus2, queen=TRUE)
cat("Summary statistics for Queen contiguity neighbors:", "\n", "\n")
```

```
## Summary statistics for Queen contiguity neighbors:
##
```

```
summary(capotuswm_q)
```

```
## Neighbour list object:
## Number of regions: 58
## Number of nonzero links: 288
## Percentage nonzero weights: 8.561237
## Average number of links: 4.965517
## Link number distribution:
##
##  2  3  4  5  6  7  8
##  2  5 15 15 16  2  3
## 2 least connected regions:
## 1679 2443 with 2 links
## 3 most connected regions:
## 315 1559 1743 with 8 links
```

```
cat("\n")
```

```
# Rook contiguity
capotuswm_r <- poly2nb(capotus2, queen=FALSE)
cat("Summary statistics for Rook contiguity neighbors:", "\n", "\n")
```

```
## Summary statistics for Rook contiguity neighbors:
##
```

```
summary(capotuswm_r)
```

```
## Neighbour list object:
## Number of regions: 58
## Number of nonzero links: 284
## Percentage nonzero weights: 8.442331
## Average number of links: 4.896552
## Link number distribution:
##
##  2  3  4  5  6  7  8
##  2  6 14 17 15  1  3
## 2 least connected regions:
## 1679 2443 with 2 links
## 3 most connected regions:
## 315 1559 1743 with 8 links
```

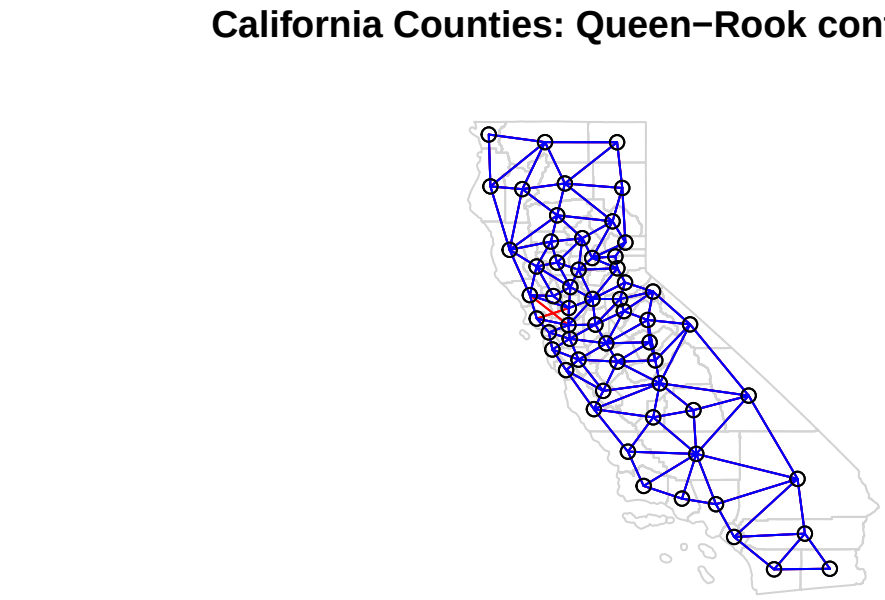
```
# Queen contiguity neighbor map
plot(capotus2, border='lightgrey')
plot(capotuswm_q, coordinates(capotus2), add=TRUE, col='red')
title(main="California Counties: Queen contiguity", cex=0.5)
```

California Counties: Queen contiguity

```
# Rook contiguity neighbor map
plot(capotus2, border='lightgrey')
plot(capotuswm_r, coordinates(capotus2), add=TRUE, col='blue')
title(main="California Counties: Rook contiguity", cex=0.5)
```

California Counties: Rook contiguity

```
# Overlay
plot(capotus2, border='lightgrey')
plot(capotuswm_q, coordinates(capotus2), add=TRUE, col='red')
plot(capotuswm_r, coordinates(capotus2), add=TRUE, col='blue')
title(main="California Counties: Queen-Rook contiguity overlay", cex=0.4)
```



[d]. Generate and plot the first-order nearest neighbors for your study region.

```
coords <- coordinates(capotus2)
head(coords)
```

```
##          [,1]     [,2]
## 8   -120.5160 39.58040
## 315 -121.3443 38.44932
## 319 -120.0555 34.69961
## 336 -120.5541 38.20461
## 382 -119.0904 34.44604
## 385 -118.2339 34.33325
```

```
k1 <- knn2nb(knearneigh(coords))
```

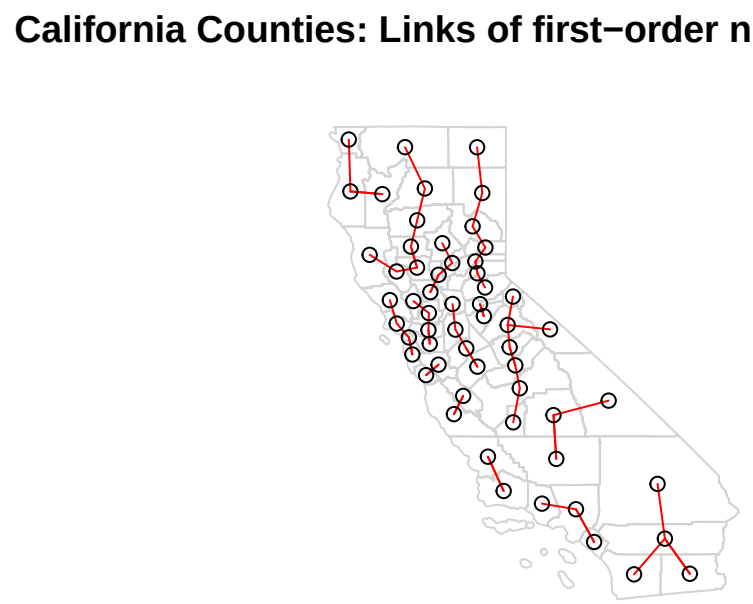
```
## Warning in knn2nb(knearneigh(coords)): neighbour object has 15 sub-graphs
```

```
k1dists <- unlist(nbdists(k1, coords, longlat=TRUE))
summary(k1dists)
```

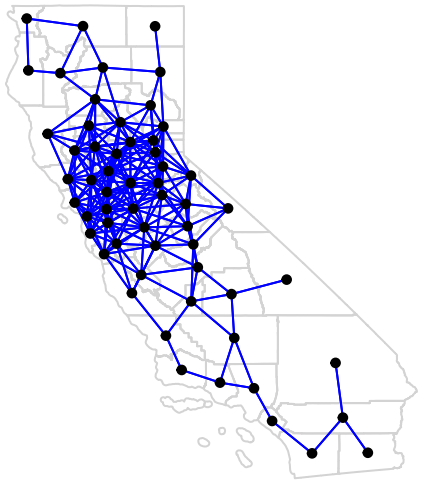
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   26.78   41.23   49.41   61.34   83.06  128.83
```

The largest nearest neighbor distance = 128.83, so we can set the upper bound for neighbors at 129

```
# Links of first-order nearest neighbors
plot(capotus2, border='lightgrey')
plot(k1, coords, add=TRUE, col='red')
title(main="California Counties: Links of first-order nearest neighbors", cex=0.5)
```




```
# Neighbors within 129km
capotuswm_d129 <- dnearneigh(coords, 0, 129, longlat=TRUE)
plot(capotus2, border='lightgrey')
plot(capotuswm_d129, coords, add=TRUE, col='blue', pch=19, cex=0.6)
title(main="California Counties: Neighbors within 129km", cex=0.5)
```



[e]. **Generate a set of distance weights for your study region and use them to identify the most accessible polygon within your study region and the most remote polygon within your study region. Explain your logic.**

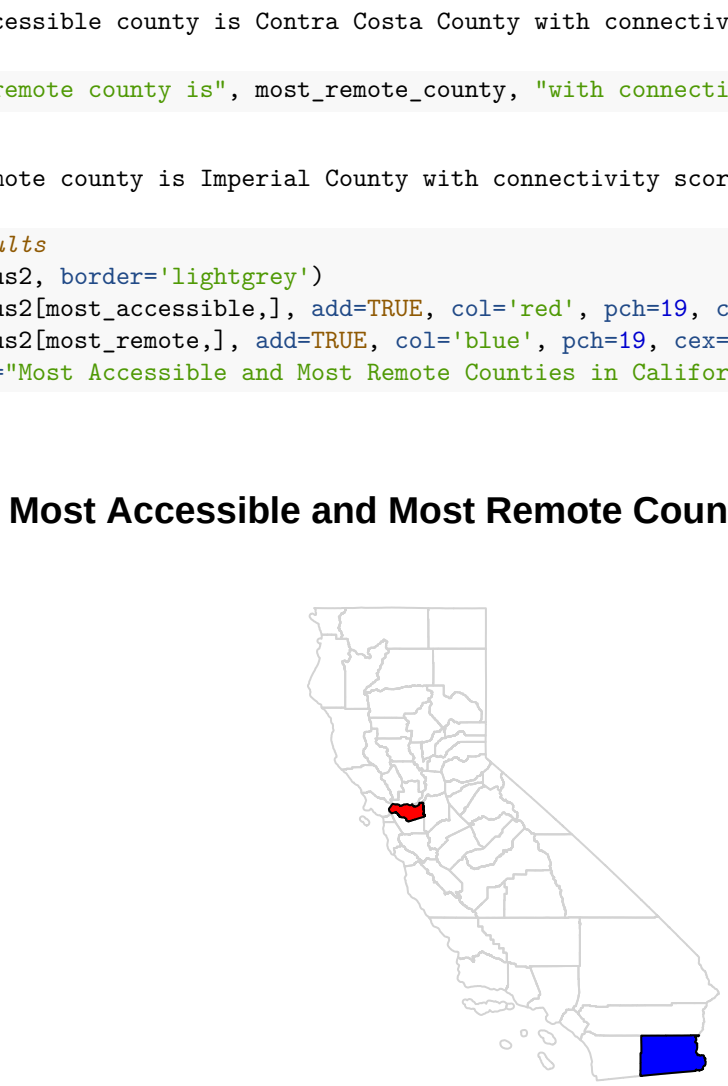
```
# Neighbor list
capotuswm_d1000 <- dnearneigh(coords, 0, 1000, longlat=TRUE)
# Inverse distance weights
dist <- nbdists(capotuswm_d1000, coords, longlat=TRUE)
idw <- lapply(dist, function(x) 1/(x))
# Neighbor list to weights list with inverse distance weights
idw_weights <- nb2listw(capotuswm_d1000, glist=idw, style="B")
# Calculate sum of weights for each polygon
conn_sums <- sapply(idw, sum)
# Find the most accessible polygon (highest connectivity)
most_accessible <- which.max(conn_sums)
most_accessible_county <- capotus$name[most_accessible]
max_conn <- conn_sums[most_accessible]
# Find the most remote polygon (lowest connectivity)
most_remote <- which.min(conn_sums)
most_remote_county <- capotus$name[most_remote]
min_conn <- conn_sums[most_remote]
# Print results
cat("Most accessible county is", most_accessible_county, "with connectivity score", max_conn, "\n")
```

```
## Most accessible county is Contra Costa County with connectivity score 0.3917643
```

```
cat("Most remote county is", most_remote_county, "with connectivity score", min_conn, "\n")
```

```
## Most remote county is Imperial County with connectivity score 0.09394591
```

```
# Plot results
plot(capotus2, border='lightgrey')
plot(capotus2[most_accessible,], add=TRUE, col='red', pch=19, cex=0.6)
plot(capotus2[most_remote,], add=TRUE, col='blue', pch=19, cex=0.6)
title(main="Most Accessible and Most Remote Counties in California", cex=0.4)
```



Explanation The identification of the Most Accessible and Most Remote counties in California is based on the sum of inverse distance weights (idw). The inverse distance weight (1/x) gives higher values to closer neighbors and lower values to distant neighbors. By summing these weights for each county, we get a measure of the counties' overall connectivity/accessibility.

The county with the highest sum of idw is considered the most accessible because it has many nearby neighbors. The county with the lowest sum of idw is considered the most remote because it has fewer nearby neighbors, or more distant neighbors. The connectivity scores presented are thus a reflection of the counties' number of connections as well as the proximity of those connections.

[2]. Using the polygon shapefile identified in question 1

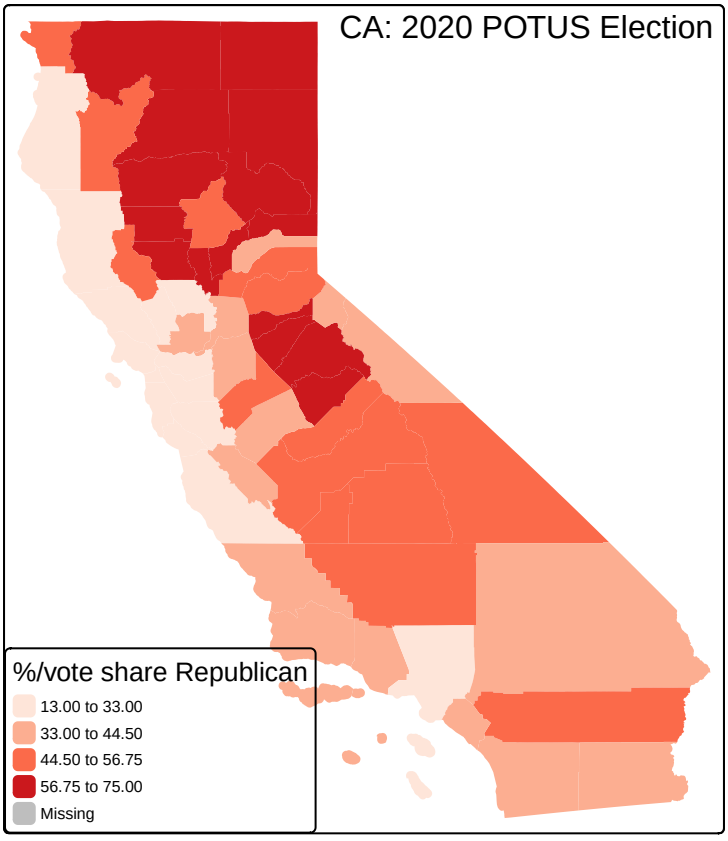
[a]. Select a variable of interest and map your variable with tmap. Show the map in your output and interpret what you see.

```
tm_shape(capotus) + tm_fill("p_R_20", style="quantile", n=4, title=("%/vote share Republican"), palette=
```

```
##

## -- tmap v3 code detected -------------------------------------------------

## [v3->v4] `tm_fill()`: instead of `style = "quantile"`, use fill.scale =
## `tm_scale_intervals()`.
## i Migrate the argument(s) 'style', 'n', 'palette' (rename to 'values') to
##   'tm_scale_intervals(<HERE>)'
## [v3->v4] `tm_fill()`: migrate the argument(s) related to the legend of the
## visual variable `fill` namely 'title' to 'fill.legend = tm_legend(<HERE>)'
## [v3->v4] `tm_layout()`: use `tm_title()` instead of `tm_layout(title = )`
## Multiple palettes called "reds" found: "brewer.reds", "matplotlib.reds". The first one, "brewer.reds"
```



Explanation The plot shows 2020 U.S. Presidential Election results for California counties. The variable mapped is 'p_R_20', which is the percentage of the vote share corresponding to Republican candidate Donald Trump. The plot shows lower vote shares for the Republican Party in metropolitan coastal regions of California, and clusters of greater support in central and northeastern California.

[b]. **Using the queen contiguity weights of question 1, generate a global Moran's I scatterplot for your variable of interest. Show and interpret the scatterplot.**

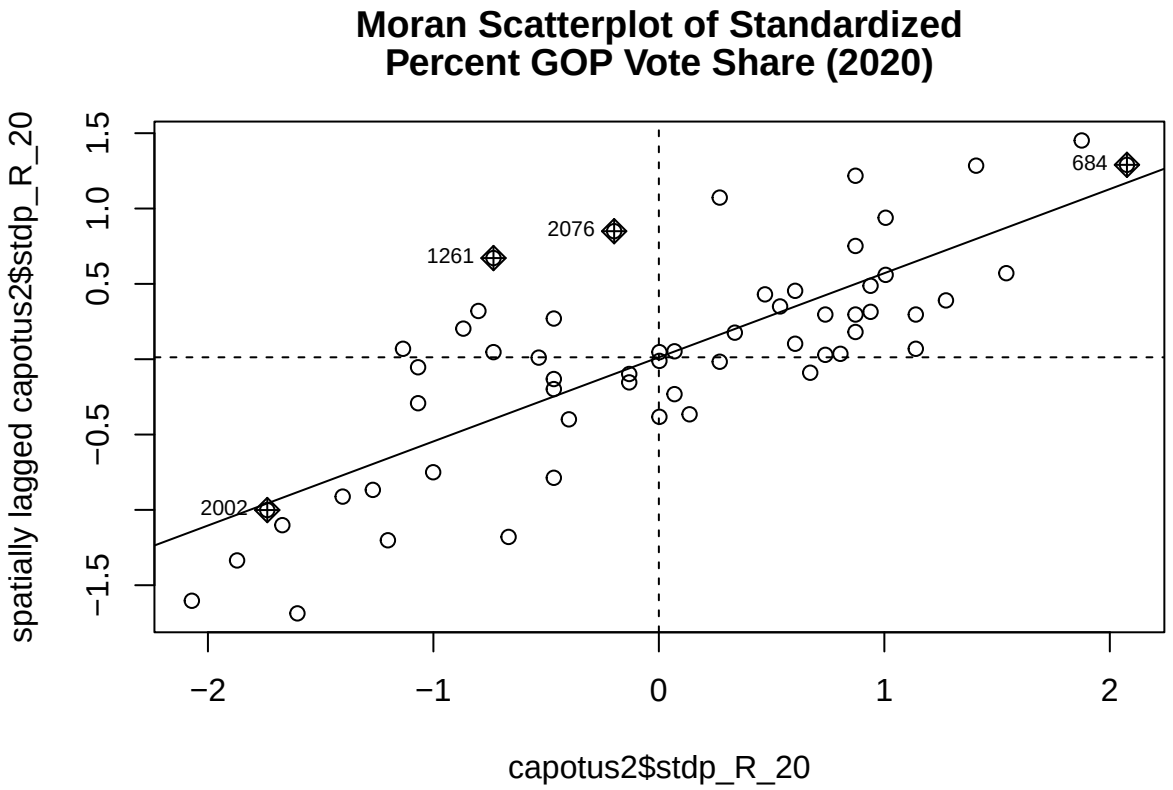
```
# Queen contiguity weights
rscapotuswm_q <- nb2listw(capotuswm_q, style="W", zero.policy=TRUE)
# Standardizing values from capotus2$p_R_20
sd(capotus2$p_R_20)
```

```
## [1] 14.949
```

```
mean(capotus2$p_R_20)
```

```
## [1] 43.96552
```

```
capotus2$stdp_R_20 <- (capotus2$p_R_20-mean(capotus2$p_R_20))/sd(capotus2$p_R_20)
# Moran scatterplot
moran.plot(capotus2$stdp_R_20, listw=rscapotuswm_q, main=c("Moran Scatterplot of Standardized", "Percen
```



Interpretation A considerable amount of points on the scatterplot fall into the 1st and 3rd quadrants, which indicates positive spatial autocorrelation. In other words, areas with similar vote shares tend to be geographically close together. The line of best fit also reflects this.

[c]. Perform a test of the hypothesis that the spatial distribution of the values of your variable of interest is random. Show your results and interpret them.

```
moran.test(capotus2$stdp_R_20, rscapotuswm_q)
```

```
##
##  Moran I test under randomisation
##
## data:  capotus2$stdp_R_20
## weights: rscapotuswm_q
##
## Moran I statistic standard deviate = 6.9872, p-value = 1.402e-12
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##       0.558221363      -0.017543860       0.006790224
```

Interpretation Moran's I statistic of 0.558 indicates considerable spatial clustering (the expected value under complete spatial randomness (CSR) is -0.018). This is reflected in the plot from [2a], from which a cursory glance reveals clustering in the Republican vote share.

The p-value of 0.000000000001402 is far less than a typical $p < 0.05$ significance threshold. Additionally, the z-score of 6.987 indicates strong statistical significance. Thus, we can confidently reject the null hypothesis of CSR.

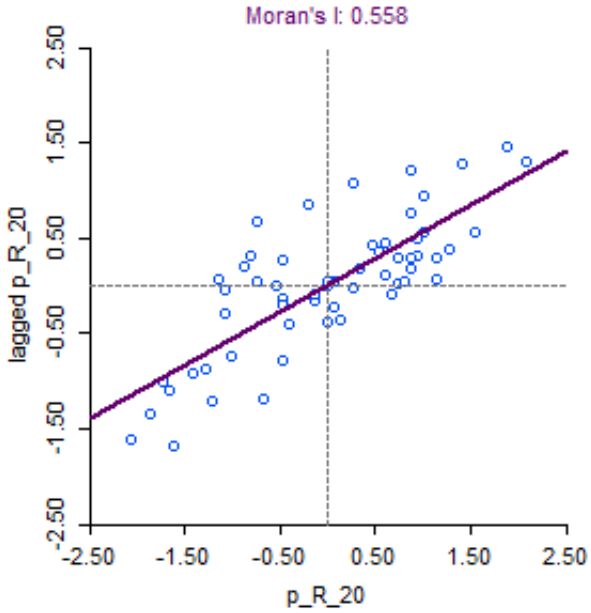
[3]. For the shape file and variable used in questions 1 and 2, read the data (shape file) into GeoDa.

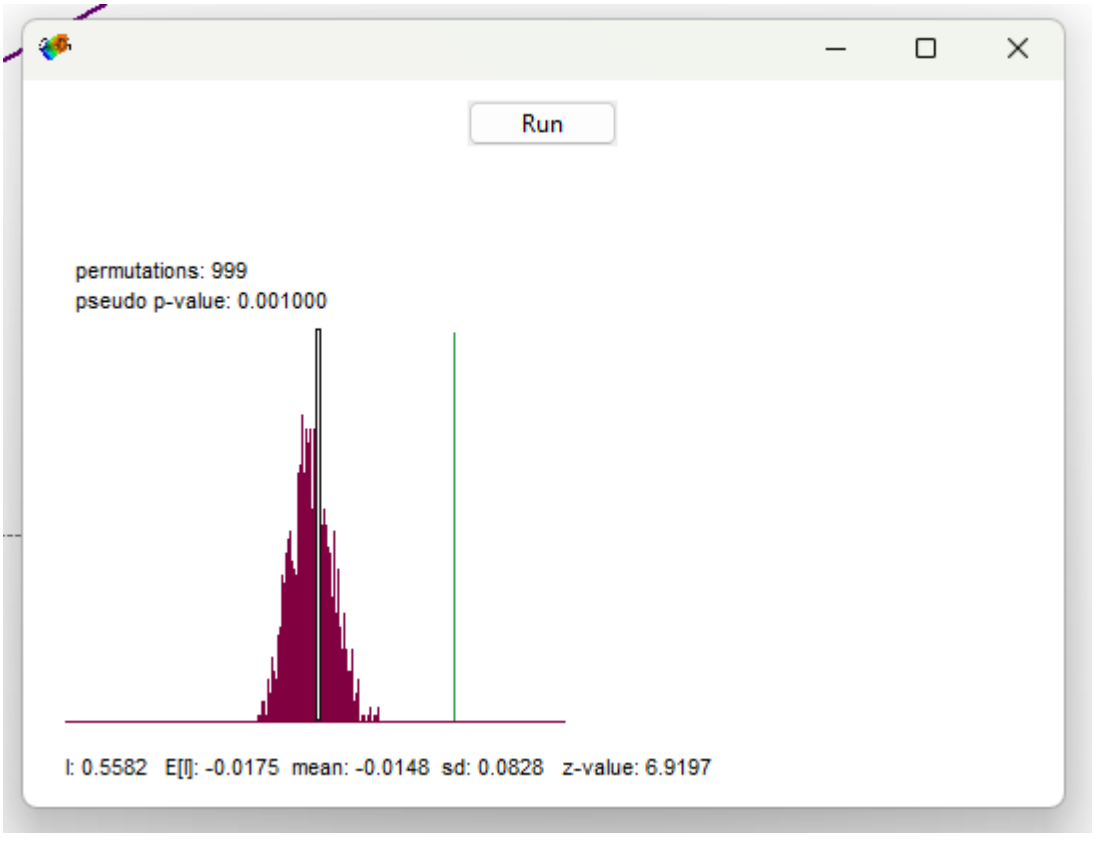
```
# st_write(capotus, "capotus.shp")
```

[a]. Generate a set of queen contiguity weights in GeoDa (order 1).

capotus.gal

[b]. Generate the global Moran's I scatterplot (again) for your variable of interest and then generate a cluster map of LISA statistics for the same variable. Show the scatterplot and LISA cluster map. For any obvious clusters, discuss the positions of the polygons in the Global Moran I plot and interpret your analysis.





The interactivity of GeoDa enables clear exploration of the spatial distribution of Republican vote share. In particular, GeoDa reveals that the cluster of 'High-High' counties on the LISA map correspond to points in

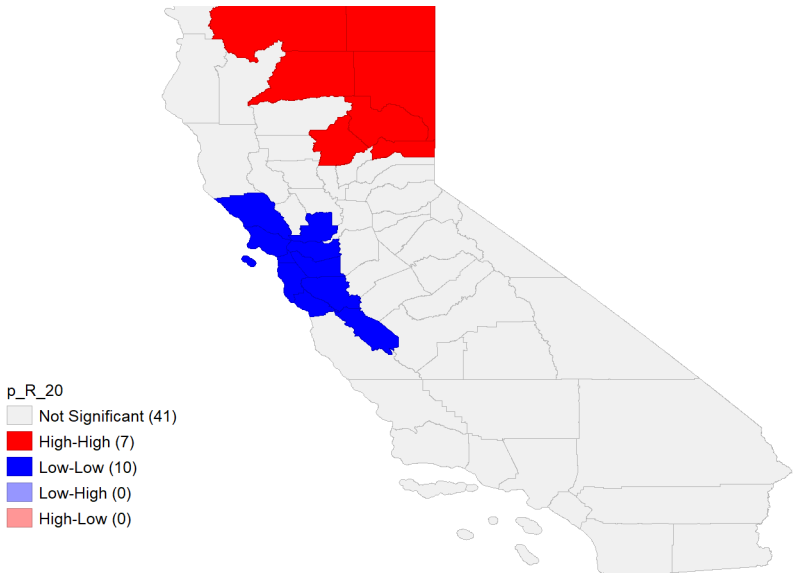


Figure 1: Local Moran's I (LISA) Cluster map.

the top right (1st quadrant) of the scatterplot. These counties had higher Republican vote shares, and their neighboring counties did as well. This resonates with our original plot from [2a], in which the counties with the greatest Republican vote share were in northeast California.

Conversely, GeoDa reveals that the cluster of 'Low-Low' counties on the LISA map correspond to points in the bottom left (3rd quadrant) of the scatterplot. These counties had remarkably low Republican vote shares, and their neighboring counties did as well. This, too, resonates with our map from [2a], which showed low Republican vote shares in San Francisco Bay counties.

[c]. Generate Getis-Ord local G-statistics (not G*), map and show the resulting clusters and relate those to your LISA plots. Are the local measures consistent with one another? If there are any significant differences, point them out and explain why they differ.

For the data I examined, there seem to be no significant differences between the results for LISA and G-statistic. The statistics themselves do differ, however; LISA identifies clusters of similar values as well as spatial outliers, while the G-statistic only identifies clusters of "highs" and clusters of "lows".

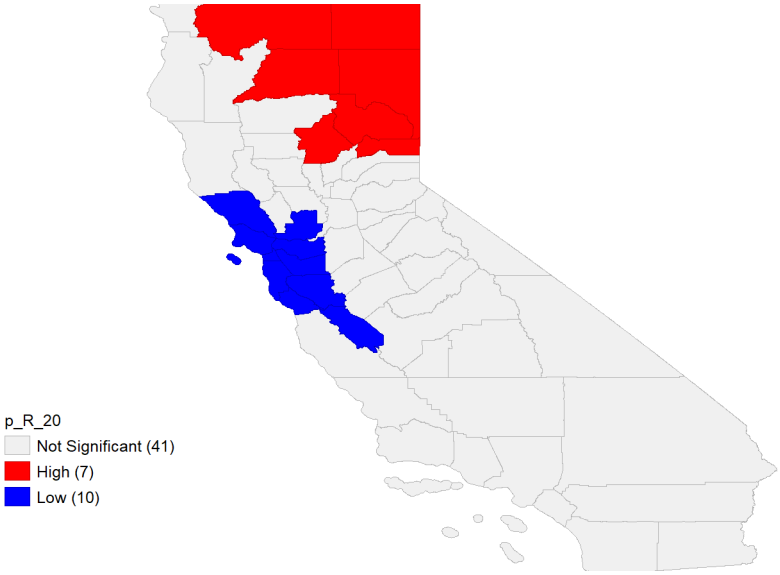


Figure 2: Local G-statistic Cluster map.