Eli Baldwin

IDCE 30274 Final

Lab Sheet

Introducing R/Rstudio with spatial vector data and analysis

Getting Started:

Open Rstudio.

Open file by going to file>open file and browsing to the correct folder, or open a new file, or both. Code can also be entered directly into the console, but we will not be doing that in this demo/lab. There should now be four “quadrants” of the Rstudio display.

**install.packages("devtools")**

**install.packages("sf")**

**install.packages("spdep")**

The first thing you want to do in R is usually install packages, which is done using the install.packages(“package”) function. One can also go to the packages tab in the bottom right quadrant. As long as you are connected to the internet you can install packages from CRAN. Packages can also be downloaded and installed from your computer. The first package I usually install is ‘devtools’, which is useful for a variety of basic R functions. The next two packages we will be installing for this demo are ‘sf’ and ‘spdep’. ‘sf’ has a variety of functions that are useful for spatial vector data. ‘spdep’ has functions for the calculation of spatial statistics such as moran’s I and was made by Luc Anselin and his posse.

While they are installing, packages that ‘sf’ and ‘spdep’ are dependent on, for example ‘rgdal’, will also be installed. In order to use functions from packages that have been installed you must use the command library(package).

**library(devtools)**

**library(sf)**

**library(spdep)**

Spatial vector operations:

Read in the shapefile, note the metadata that is printed out. st\_read is an ‘sf’ function.

**test <- st\_read("Z:/personal/ebaldwin/RStudio\_Demo/ConCounties2.shp")**

test is now an object of the type sf. Note that a new row has appeared in the tope right quadrant under data that says ‘test 3108 obs. Of 35 variables.

You can click on the spreadsheet icon next to the variable ‘test’ on the top right quadrant or type view(test) in the console to view the data. This will bring up a new tab in the top left quadrant.

The metadata indicates that the data is currently in a latlong projection, lets change that to albers equal area conic.

#project the data to the albers equal area north american projection. This style is proj4string. proj4string is used by ‘rgdal’, I got this string from ESRI.

**test\_proj <- st\_transform(test, "+proj=aea +lat\_1=20 +lat\_2=60 +lat\_0=40 +lon\_0=-96 +x\_0=0 +y\_0=0 +ellps=GRS80 +datum=NAD83 +units=m +no\_defs")**

#convert the polygons to points by centroid.

**test\_points <- st\_centroid(test\_proj)**

Spatial Statistics Operations:

We are now moving on to using functions from the spdep package. The next three lines of code are all to create spatial weights based on the k-nearest neighbors conceptualization of spatial relationships. This a lengthier process than in ArcMap or Geoda.

#calculate 15 nearest neighbors for each point.

**test\_knn <- knearneigh(test\_points, k=15)**

#create a variable that is the neighbors of each feature based on knn. nb is a neighbors list

**test\_nb <- knn2nb(test\_knn)**

#turn the neighbors object into a weights list.

**test\_listw <- nb2listw(test\_nb)**

We now need to break out the life expectancy column from the sf object. The function as.vector coerces a set of values into vector format. Datatable$column references one column within the data table (usually a data frame, but in this case an sf object).

#turn the column from the data that is life expectancy by county into a stand-alone variable

**test\_LE <- as.vector(test\_proj$Lifeexpect)**

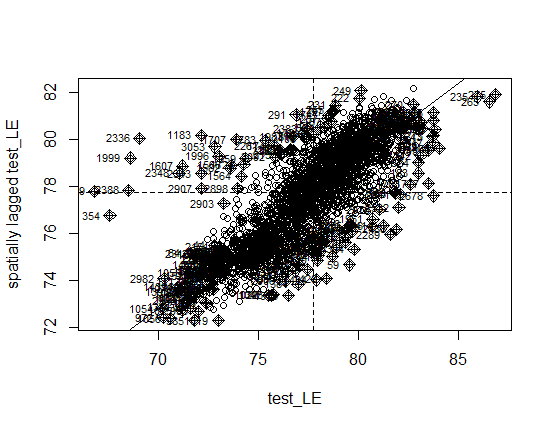
#Run a moran's I test along with calculation of statistical significance using 499 permutations

**moran.mc(test\_LE, test\_listw, nsim = 499)**

#plot the graph of moran's I

**moran.plot(test\_LE, test\_listw)**

notice that the graph appears in the plot tab in the lower right-hand quadrant.



#calculate local G, aka Getis-Ord, aka hot spot analysis

**test\_G <- localG(test\_LE, test\_listw)**

‘test\_G’ is now an object of the type ‘localG’

#turn 'test\_G' into vector (not spatial) variable

**test\_G <- as.vector(test\_G)**

R data manipulation operations:

Now, we want to take the z-scores output by localG, and turn them into something useful: hot and cold spot designations. In order to iterate through each value in one vector or data frame I recommend the use of ‘sapply’. The apply family of functions are considered better than for loops because they are less likely to crash the program. ‘sapply’ has a function defined within it. ‘sapply’ uses parentheses. Functions and if statements use parentheses for the arguments and curly brackets to bound the main sections.

#create new vector variable to divide all counties into hot, cold, or neither

**hot\_cold <- sapply(test\_G, function(x){**

#the value of 3.886 is the critical (z-score) value for statistical significance at a 95% confidence level for over 1000 observations according to the localG documentation

**if (x > 3.886){**

**x = "Hot"**

**} else if (x < -3.886){**

**x = "Cold"**

**} else {**

**x = "Neither"**

**}**

**})**

the function cbind is used to add a new column of already existing data to a data table (usually a data frame).

#bind the new column to the data

**test\_proj <- cbind(test\_proj, hot\_cold)**

We will now create a simple map of hot and cold spots. I tried using the default palette and did not like it. I am unsure why cold, hot and neither were put in that order for the legend, but they were. So, we must make the palette in the same order. The function c() is used to assign a variable a series of values.

#create palette for map display

**palette = c("blue", "red", "white")**

Mapping operation:

The function plot can take a variety of inputs, but can also be run with only the data to be used specified, in this case the hot\_cold column of the test\_proj sf object. The argument ‘main’ refers to the title, and ‘pal’ to the palette. For some reason the column index call is different from usual.

#plot the hot and cold spots

**plot (test\_proj["hot\_cold"], main = "Hot/Cold spots of Life Expectancy", pal = palette)**

This plot can be saved by going to, on the top, plots>save as image… or the export button in the plot quadrant.

