Spatial Autocorrelation

First, a note on spatial data

- **Point data**
  - Accuracy of location is very important

- **Area/lattice data**
  - Data reported for some regular or irregular areal unit

- **2 key components of spatial data:**
  - Attribute data
  - Spatial data

- **How you choose to sample, or aggregate, your data is very important**
  - Affect spatial relationships in your data
Problems/challenges with areal data

- Modifiable areal unit problem (MAUP)
  - Scale effect – spatial data analysis at different scales may produce different results
  - Zoning effect – regrouping zones at a given scale may produce different results

- What are the???
  - Optimal neighborhood/area size
  - Alternative zoning schemes

Spatial Autoc. is Scale-Dependent!
Tobler’s First Law of Geography

- “All places are related but nearby places are more related than distant places”

- Spatial autocorrelation is the formal property that measures the degree to which near and distant things are related
  - Statistical test of match between locational similarity and attribute similarity
    - Positive, negative or zero relationship
### Spatial Autocorrelation

- **Null hypothesis:**
  - Spatial randomness
  - Values observed at one location do not depend on values observed at neighboring locations
  - Observed spatial pattern of values is equally likely as any other spatial pattern
  - The location of values may be altered without affecting the information content of the data

<table>
<thead>
<tr>
<th>Regular</th>
<th>Random</th>
<th>Aggregated</th>
</tr>
</thead>
</table>

### Spatial autocorrelation

Steps in determining the extent of spatial autocorrelation in your data:

1. **Choose a neighborhood criterion**
   - Which areas are linked?
2. **Assign weights to the areas that are linked**
   - Create a spatial weights matrix
3. **Run statistical test, using weights matrix, to examine spatial autocorrelation**
Spatial weights matrices

- Neighborhoods can be defined in a number of ways
  - Contiguity (common boundary)
    - What is a “shared” boundary?
  - Distance (distance band, K-nearest neighbors)
    - How many “neighbors” to include, what distance do we use?
  - General weights (social distance, distance decay)

![Spatial weights matrices diagram](image)

Common weights measures

- Most common is using binary connectivity based on contiguity
  - $w_{ij} = 1$ if regions $i$ and $j$ are contiguous, $w_{ij} = 0$ otherwise

- May also be defined as a function of the distance between $i$ and $j$
  - Distance of the line connecting the centroids of two areas
    - $w_{ij} = d_{ij}^{-\beta}$
    - $w_{ij} = \exp[-\beta d_{ij}]$
Example: contiguity weights

\[
\begin{matrix}
We & Ad & Ar & De & Bo & Je & Do \\
We & 1 & 1 & 0 & 0 & 1 & 0 \\
Ad & 0 & 1 & 1 & 0 & 1 & 1 \\
Ar & 0 & 1 & 1 & 1 & 0 & 1 \\
De & 0 & 1 & 1 & 1 & 0 & 1 \\
Bo & 0 & 1 & 1 & 1 & 0 & 1 \\
Je & 0 & 1 & 1 & 1 & 1 & 0 \\
Do & 0 & 0 & 1 & 0 & 0 & 1 \\
\end{matrix}
\]

Step 1: Choose a neighborhood criterion

Importing shapefiles into R and constructing neighborhood sets
R libraries we’ll use

SET YOUR CRAN MIRROR

> install.packages("ctv")  
> library("ctv")  
> install.views("Spatial")

You only need to do this once on your (or KESDA) computers

> library(maptools)  
> library(rgdal)  
> library(spdep)

Importing a shapefile

> library(maptools)

> getinfo.shape("F:/R/shapefiles/sids2.shp")
Shapefile type: Polygon, (5), # of Shapes: 100

> sids<-readShapePoly("F:/R/shapefiles/sids2.shp")

> class(sids)
[1] "SpatialPolygonsDataFrame"
attr("package")
Importing a shapefile (2)

```r
library(rgdal)

sids <- readOGR(dsn="F:/R/shapefiles",layer="sids2")
OGR data source with driver: ESRI Shapefile
Source: " F:/R/shapefiles", layer: "sids2"
with 100 features and 18 fields
Feature type: wkbPolygon with 2 dimensions

class(sids)
[1] "SpatialPolygonsDataFrame"
attr("package")
[1] "sp"
```

Projecting a shapefile

- If the shapefile has no .prj file associated with it, you need to assign a coordinate system

```r
proj4string(sids) <- CRS("+proj=longlat ellps=WGS84")
```

- We can then transform the map into any projection

```r
sids_NAD <- spTransform(sids, CRS("+init=epsg:3358"))

sids_SP <- spTransform(sids, CRS("+init=ESRI:102719"))
```

- For a list of applicable CRS codes:
  - [http://www.spatialreference.org/ref/](http://www.spatialreference.org/ref/)
    - Stick with the epsg and esri codes
Contiguity based neighbors

- Counties sharing any boundary point (QUEEN) are taken as neighbors, using the `poly2nb` function, which accepts a `SpatialPolygonsDataFrame`
  ```r
  library(spdep)
  sids_nbq <- poly2nb(sids)
  ```

- If contiguity is defined as counties sharing more than one boundary point (ROOK), the `queen=` argument is set to `FALSE`
  ```r
  sids_nbr <- poly2nb(sids, queen=FALSE)
  ```
  ```r
  coords <- coordinates(sids)
  plot(sids)
  plot(sids_nbq, coords, add=T)
  ```
Distance based neighbors
k nearest neighbors

- Can also choose the k nearest points as neighbors

```r
> coords<-coordinates(sids_SP)
> IDs<-row.names(as(sids_SP, "data.frame"))

> sids_kn1<-knn2nb(knearneigh(coords, k=1), row.names=IDs)
> sids_kn2<-knn2nb(knearneigh(coords, k=2), row.names=IDs)
> sids_kn4<-knn2nb(knearneigh(coords, k=4), row.names=IDs)

> plot(sids_SP)
> plot(sids_kn2, coords, add=T)
```
Can I do that with point data?

```r
bost <- read.csv("C:/Users/ERoot/Desktop/R/boston.csv", sep="", header=T)
b.coord <- SpatialPoints(bost[,c("LON","LAT")])
bost2 <- SpatialPointsDataFrame(b.coord, bost)

coord_b <- coordinates(bost2)
class(coord_b)
[1] "matrix"

bost_k2 <- knn2nb(knearneigh(coord_b, k=2, longlat=T))

plot(as(bost2,"Spatial"), axes=T)
plot(bost_k2, coord_b, add=T)
plot(bost2[bost2$CHAS==1,], col="blue", add=TRUE)
```
Distance based neighbors
Specified distance

- Can also assign neighbors based on a specified distance

```r
> dist <- unlist(nbdists(sids_kn1, coords))
> summary(dist)

  Min.  1st Qu.   Median     Mean  3rd Qu.     Max. 
  40100  89770  97640  96290 107200  134600
> max_k1 <- max(dist)

> sids_kdl <- dnearneigh(coords, d1=0, d2=0.75*max_k1, row.names=IDs)
> sids_kd2 <- dnearneigh(coords, d1=0, d2=1*max_k1, row.names=IDs)
> sids_kd3 <- dnearneigh(coords, d1=0, d2=1.5*max_k1, row.names=IDs)
```

**OR by raw distance**

```r
> sids_ran1 <- dnearneigh(coords, d1=0, d2=134600, row.names=IDs)
```
Step 2: Assign weights to the areas that are linked

Creating spatial weights matrices using neighborhood lists
Spatial weights matrices

- Once our list of neighbors has been created, we assign spatial weights to each relationship
  - Can be binary or variable
  - If we don’t know much about the spatial process, try to stick with binary weights
  - Even when the values are binary 0/1, the issue of what to do with no-neighbor observations arises

- Binary weighting will, for a target feature, assign a value of 1 to neighboring features and 0 to all other features
  - Used with fixed distance, k nearest neighbors, and contiguity

Row-standardized weights matrix

```
> sids_nbq_w<- nb2listw(sids_nbq)
> sids_nbq_w
```

Characteristics of weights list:
- Neighbour list object:
  - Number of regions: 100
  - Number of nonzero links: 490
  - Percentage nonzero weights: 4.9
  - Average number of links: 4.9

Weights style: W
Weights constants summary:
```
   n  nn  S0   S1  S2
   W 100 10000 100 44.65023 410.4746
```

- Row standardization is used to create proportional weights in cases where features have an unequal number of neighbors
  - Divide each neighbor weight for a feature by the sum of all neighbor weights
    - Obs i has 3 neighbors, each has a weight of 1/3
    - Obs j has 2 neighbors, each has a weight of 1/2
  - Use is you want comparable spatial parameters across different data sets with different connectivity structures
Binary weights

```r
> sids_nbq_wb <- nb2listw(sids_nbq, style="B")
> sids_nbq_wb
```

Characteristics of weights list:
Neighbour list object:
Number of regions: 100
Number of nonzero links: 490
Percentage nonzero weights: 4.9
Average number of links: 4.9

Weights style: B
Weights constants summary:
```
 n  nn  S0  S1  S2
 B 100 10000 490 980 10696
```

- Row-standardised weights increase the influence of links from observations with few neighbours
- Binary weights vary the influence of observations
  - Those with many neighbours are up-weighted compared to those with few

Binary vs. row-standardized

- A binary weights matrix looks like:
  ```
  0 1 0 0
  0 0 1 1
  1 1 0 0
  0 1 1 1
  ```
  - Observation 1 has neighbor 2
  - Observation 2 has neighbors 3 and 4
  - Observation 3 has neighbors 1 and 2
  - Observation 4 has neighbor 2, 3 and 4

- A row-standardized matrix it looks like:
  ```
  0 1 0 0
  0 0 .5 .5
  .5 .5 0 0
  0 .33 .33 .33
  ```
  - Observation 1 has neighbor 2
  - Observation 2 has neighbors 3 and 4
  - Observation 3 has neighbors 1 and 2
  - Observation 4 has neighbor 2, 3 and 4
Regions with no neighbors

- If you ever get the following error:

```
Error in nb2listw(filename): Empty neighbor sets found
```

- You have some regions that have NO neighbors
  - This is most likely an artifact of your GIS data (digitizing errors, slivers, etc), which you should fix in a GIS
  - Also could have “true” islands (e.g., Hawaii, San Juans in WA)
    - May want to use k nearest neighbors
    - Or add zero.policy=T to the nb2listw call

```r
> sids_nbq_w<-nb2listw(sids_nbq, zero.policy=T)
```

Weights based on IDW

```r
> dist<-nbdists(sids_nbq, coordinates(sids_SP))
> idw<-lapply(dist, function(x) 1/(x/1000))

> sids_nbq_idwb<-nb2listw(sids_nbq, glist=idw, style="B")

> summary(unlist(sids_nbq_idwb$weights))

       Min.   1st Qu.    Median      Mean   3rd Qu.      Max.      
0.004123 0.006274 0.007640  0.008037 0.009268  0.024930
```
Step 3: Examine spatial autocorrelation

Using spatial weights matrices, run statistical tests of spatial autocorrelation

Spatial autocorrelation

- Test for the presence of spatial autocorrelation
  - Global
    - Moran’s I
    - Geary’s C
  - Local (LISA – Local Indicators of Spatial Autocorrelation)
    - Local Moran’s I and Getis $G_i^*$

- Other tests that are more simple:
  - The Chi-square Test for Spatial Independence
  - The Join Count Statistic
Moran’s I

- Classic/best measure of spatial autocorrelation
- Depends upon definition of neighboring unit via the spatial weights matrix
- Typically ranges from -1 to 1

- Like regression, it has a few assumptions
  - Regional x/y values all come from normal distributions w/same mean and variance for each region
  - Randomly rearrange the data on map and compute I many times, would have a normal distribution
    - Why? Because we use the normal distribution to calculate the p-value

\[ I = \frac{n \sum_{i=1}^{n} \sum_{i \neq j}^{n} w_{ij} (y_i - \bar{y})(y_j - \bar{y})}{\left( \sum_{i=1}^{n} (y_i - \bar{y})^2 \right) \cdot \left( \sum_{i \neq j} w_{ij} \right)} \]

- \( n \) = number of regions
- \( w_{ij} \) = measure of spatial proximity between region i and j

Product of the deviation from the mean for all pairs of adjacent regions (\( w_{ij} = 1 \))

Essentially a measure of variance across the regions

Sum of the weights (count of all adjacent pairs)
Moran’s I in R

> moran.test(sids_NAD$SIDR79, listw=sids_nbq_w, alternative="two.sided")

Moran's I test under randomisation

data:  sids_NAD$SIDR79
weights: sids_nbq_w

Moran I statistic standard deviate = 2.3625, p-value = 0.009075
alternative hypothesis: greater
sample estimates:
Moran I statistic       Expectation          Variance
0.142750392             -0.010101010       0.004185853

Moran’s I in R

> moran.test(sids_NAD$SIDR79, listw=sids_nbq_wb)

Moran's I test under randomisation

data:  sids_NAD$SIDR79
weights: sids_nbq_wb

Moran I statistic standard deviate = 1.9633, p-value = 0.02480
alternative hypothesis: greater
sample estimates:
Moran I statistic       Expectation          Variance
0.110520684             -0.010101010       0.003774597
What if my data violate the assumptions?

If you doubt that the assumptions of Moran’s I are true (normality and randomization), we can use a Monte Carlo simulation

- Simulate Moran’s I n times under the assumption of no spatial pattern
  - Assigning all regions the mean value
  - Calculate Moran’s I
- Compare actual value of Moran’s I to randomly simulated distribution to obtain p-value

Monte Carlo Moran’s I

```r
> set.seed(1234)
> bperm<-moran.mc(sids_NAD$SIDR79,listw=sids_nbq_w,nsim=999)
> bperm

Monte-Carlo simulation of Moran's I

data:  sids_NAD$SIDR79
weights: sids_nbq_w
number of simulations + 1: 1000

statistic = 0.1428, observed rank = 991, p-value = 0.009
alternative hypothesis: greater
```
Monte Carlo Moran’s I

> mean(bperm$res[1:999])
[1] -0.0130007
> var(bperm$res[1:999])
[1] 0.004201103
> summary(bperm$res[1:999])
Min, Max, Median, Quartiles

> hist(bperm$res, freq=TRUE, breaks=20, xlab="Simulated Moran’s I")
> abline(v=0, col="red")

Geary’s C

- Geary’s C typically ranges from 0 to 3
  - Cannot be negative
- An uncorrelated process has an expected C = 1
  - Values less than 1 indicate positive spatial autocorrelation
  - Values greater than 1 indicate negative autocorrelation

\[ I = \frac{(n-1) \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} (y_i - y_j)^2}{2 \left( \sum_{i=1}^{n} (y_i - \bar{y})^2 \right) \left( \sum_{i \neq j} w_{ij} \right)} \]
Relationship of Moran’s I and Geary’s C

- C approaches 0 and I approaches 1 when similar values are clustered
- C approaches 3 and I approaches -1 when dissimilar values tend to cluster
- High values of C measures correspond to low values of I
- So the two measures are inversely related

Geary’s C in R

```R
> geary.test(sids_NAD$SIDR79, listw=sids_nbq_w)

Geary's C test under randomisation

data:  sids_NAD$SIDR79
weights: sids_nbq_w

Geary C statistic standard deviate = 2.5449, p-value = 0.005466
alternative hypothesis: Expectation greater than statistic
sample estimates:
Geary C statistic       Expectation          Variance
 0.819475730           1.000000000       0.005031937
```
Geary's C in R

```r
> geary.test(sids_NAD$SIDR79, listw=sids_nbq_wb)

Geary's C test under randomisation

data: sids_NAD$SIDR79
weights: sids_nbq_wb

Geary C statistic standard deviate = 2.5788, p-value = 0.004957
alternative hypothesis: Expectation greater than statistic
sample estimates:
Geary C statistic       Expectation          Variance
0.773201052       1.000000000       0.007734876
```

Correlogram of spatial lags

```r
> cor8<sp.correlogram(sids_nbq, sids_NAD$SIDR79, order=8, method="I", style="W")
> print(cor8)

Spatial correlogram for sids_NAD$SIDR79
method: Moran's I

<table>
<thead>
<tr>
<th></th>
<th>estimate</th>
<th>expectation</th>
<th>variance</th>
<th>standard deviate</th>
<th>Pr(I) two sided</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.1427504</td>
<td>-0.0101010</td>
<td>0.0041859</td>
<td>2.3625</td>
<td>0.018151 *</td>
</tr>
<tr>
<td>2</td>
<td>0.1268861</td>
<td>-0.0101010</td>
<td>0.0018814</td>
<td>-2.6924</td>
<td>0.007094 **</td>
</tr>
<tr>
<td>3</td>
<td>0.1268861</td>
<td>-0.0101010</td>
<td>0.0018814</td>
<td>-2.6924</td>
<td>0.007094 **</td>
</tr>
<tr>
<td>4</td>
<td>0.0013193</td>
<td>0.017049</td>
<td>0.0017511</td>
<td>0.2148</td>
<td>0.829998</td>
</tr>
<tr>
<td>5</td>
<td>0.0378019</td>
<td>-0.0101010</td>
<td>0.0019625</td>
<td>-0.2147</td>
<td>0.829998</td>
</tr>
<tr>
<td>6</td>
<td>0.0818992</td>
<td>0.0013193</td>
<td>0.0022660</td>
<td>-1.5083</td>
<td>0.131484</td>
</tr>
<tr>
<td>7</td>
<td>0.0294425</td>
<td>0.0017511</td>
<td>0.0025599</td>
<td>0.7816</td>
<td>0.434472</td>
</tr>
</tbody>
</table>

---
Signif. codes:  0 ****  0.001 ***  0.01 **  0.05 *  0.1 .  1

> plot(cor8)
```
Global vs. Local Analysis

- **Global**
  - One statistic to summarize pattern in whole study area
  - Clustering
  - Homogeneity
- **LISA: local indicators of spatial association**
  - Location-specific statistics
  - Clusters/hot-spots
  - Heterogeneity
Local Moran’s I

- Used to determine if local autocorrelation exists around each region
- Returns an I value for each region
- Often used after global Moran’s I to see if:
  - The study area is homogeneous (local statistics similar across regions)
  - There are local outliers that contribute to a significant global statistic

\[ I_i = \frac{n(y_i - \bar{y})}{\sum_j (y_i - \bar{y})^2} \sum_j w_{ij} (y_j - \bar{y}) \]

Exporting local I for ArcGIS

- Run local Moran’s I and assign FIPS code to each
  > fips <- order(sids_NAD$FIPSNO)
  > nclocI <- localmoran(sids_NAD$SIDR79, sids_nbq_w)
  > printCoefmat(data.frame(nclocI[fips,], row.names=sids_NAD$FIPSNO[fips]), check.names=FALSE)

- Create a dataframe with the Moran’s output and export to a .csv file
  > lmi<-data.frame(resI[oid,], row.names=sids_NAD$FIPSNO[oid])
  > write.table(lmi, file="C:/Users/Elisabeth Root/Desktop/Quant/R/lmi.csv", sep = "",)
Local Moran’s I in R

```r
> nci <- moran.plot(sids_NAD$SIDR79, sids_nbq_w, labels=as.character(sids_NAD$NAME), xlim=c(-1,6.5), ylim=c(-1,4.5), xlab="SIDS Rate", ylab="SL SIDS Rate")
```

- Plots local Moran’s I values and highlights statistically significant outliers (p<0.05)

Mapping local outliers

```r
> infl <- apply(nci$is.inf, 1, any)
> x <- sids_NAD$SIDR79
> lhx <- cut(x, breaks=c(min(x), mean(x), max(x)), labels=c("L","H"), include.lowest=TRUE)
> wx <- lag(sids_nbq_w, sids_NAD$SIDR79)
> lhwx <- cut(wx, breaks=c(min(wx), mean(wx), max(wx)), labels=c("L","H"), include.lowest=TRUE)
> lhlh <- interaction(lhx, lhwx, infl, drop=TRUE)
> cols <- rep(1, length(lhlh))
> cols[lhlh == "H.L.TRUE"] <- 2
> cols[lhlh == "L.H.TRUE"] <- 3
> cols[lhlh == "H.H.TRUE"] <- 4
> plot(sids_NAD, col=grey.colors(4, 0.95, 0.55, 2.2)[cols])
> legend("topright", legend=c("None", "HL", "LH", "HH"), fill=grey.colors(4, 0.95, 0.55, 2.2), bty="n", cex=0.8, y.intersp=0.8)
```
Mapping local outliers

Other functions used for datasets with few neighbors:
localmoran.sad()
localmoran.exact()

Next Step?
Using measures of SA in regression

1. Given a problem with a spatial dimension, estimate parameters using OLS
2. Evaluate the model – are assumptions of OLS met?
3. Construct spatial weights matrix
4. Test for spatial dependence in the residuals
5. Modify functional form as necessary
   ▶ Spatial lag, spatial error, GWR