Lab 5: Lattice Data Analysis

Objectives

- Conduct ESDA using lattice data
- Fit spatial regression models

Part I: ESDA Using R

1. Load necessary R packages

We will use two packages in this lab: `spdep` and `rgdal`.

```
> install.packages("spdep",repos = "http://cran.r-project.org")
> install.packages('RSEIS') # matrix manipulation
> install.packages('RColorBrewer')
> install.packages('spatialreg')
> library(spdep)
> library(rgdal)   # we used this in Lab 1
> library('RSEIS')
> library(RColorBrewer)
> library(spatialreg)
```

Suppose you have already created a folder named as Geog8102 under c:\WorkSpace\, i.e., you have the folder: "`c:\WorkSpace\Geog8102`". You can replace Geog8102 with any name you like.

Suppose the lab materials have been downloaded and unzipped to `C:\WorkSpace\Geog8102\Lab5`. In the folder `C:\WorkSpace\Geog8102\Lab5`, there is a folder named "data". That is where the working data are stored. You can set your working directory to the data folder, i.e.,
`C:\WorkSpace\Geog8102\Lab5\data\`.

```
> setwd("c:/WorkSpace/Geog8102/Lab5/data")   # set working directory
```

2. Read data

Recall that in the exercise of Lab 1, we explored the shapefile Columbus dataset. We return to this dataset and do some spatial analysis. First, we read the data into R.

```
> ColData = readOGR(".", "columbus")
> coords = coordinates(ColData)      # coordinates of the census tract centroids
> IDs = row.names(as(ColData, "data.frame"))
```

3. Neighborhood

(1) Adjacency connections

In spatial analysis of lattice data, neighborhood is an important component. It is usually used to generate spatial weight. One way to define neighbors is based on the adjacency connection. This will generate the links between each areal unit (e.g., census tract) and its adjacent areal units (e.g., adjacent census tracts).

We will use the `poly2nb` function in `spdep` library. This function builds a neighbors list based on regions with contiguous boundaries, which are sharing one or more boundary point.

```
> col_nb = poly2nb(ColData, queen = TRUE)
> summary(col_nb)
> class(col_nb)      # check the class of col_nb
> help(poly2nb)       # check help to see detailed information about poly2nb function
```

We can plot a bar plot similar to the clustered column chart using Microsoft Excel. The horizontal axis is the number of neighbors, and the vertical bars are the number of census tracts that have the corresponding number of neighbors. To do so, we will call function `unlist()` to simply `col_nb` to a vector. Then we use table function to convert the results into a frequency table.

```
> nb = unlist(col_nb)
> class(nb)      # check the class of nb
> nb
> nbc1 = table(nb)     #table shows the number of neighbors for each polygon
> nbc2 = table(nbc1)  #table shows the distribution of neighbor connections
                        #i.e., the number of cases that have certain number of neighbors
> nbc1
> nbc2
> barplot(nbc2,col=rainbow(20), xlab="number of neighbors",ylab="number of
cases")
```

Assignment I
o Provide the barplot in your report.
o Briefly interpret the connectivity of Columbus census tracts based on the plot.

We can also plot the spatial adjacency links with the census tracts.

```
> plot(ColData, border = "grey60")
> plot(col_nb, coords, pch = 19, cex = 0.6, add = TRUE)
> title("Adjacency Connection")
> box()
```

(2) K nearest neighbors

Another way is to build connection based on *k*-nearest neighbors. For example, we can build the 1, 2 nearest neighbors connection as follows.

```
> nbk1 = knn2nb(knearneigh(coords, k=1), row.names=IDs)
> nbk2 = knn2nb(knearneigh(coords, k=2), row.names=IDs)
```

Then we can plot the connections based on k nearest neighbors.

```
> dev.off()          # close previous plot if there is any
> dev.new(width=8, height=4)
> par(mfrow = c(1, 2), mar=c(1,1,1,1), pty = "s")

> plot(ColData, border="grey60")
> plot(nbk1, coords, add=TRUE, pch=19, cex=0.6)
> text(bbox(ColData)[1,1] + 0.5, bbox(ColData)[2,2], labels="k=1")
> box()

> plot(ColData, border="grey60")
> plot(nbk2, coords, add=TRUE, pch=19, cex=0.6)
> text(bbox(ColData)[1,1] + 0.5, bbox(ColData)[2,2], labels="k=2")
> box()

> par(mfrow=c(1,1))
```

Assignment II
o Provide the plot for 1, 2 nearest neighbors in your report.
o Ask yourself how to interpret the plot. (You do not need to report your interpretation)

(3) Distance-based neighbors

We can also build neighborhood connection based on distance. First, we need calculate the average distance for adjacent neighbors so that we can specify the neighborhood distance threshold. We will use function `nbdists` to calculate the Euclidean distances along the neighborhood links.

```
> dsts = unlist(nbdists(col_nb, coords))
> summary(dsts)
> hist(dsts)
```

The mean distance is 0.4754 (degrees). Therefore, we can specify distance by 0.5, 0.8 to get different level of connection.

```
> dist1=0.5
> dist2=0.8

> nb1 = dnearneigh(coords, d1=0, d2=dist1, row.names=IDs)
> nb2 = dnearneigh(coords, d1=0, d2=dist2, row.names=IDs)
```

Now we plot the connections with census tract polygons.

```
> dev.off()
> dev.new(width=8, height=4)
> par(mfrow = c(1, 2), mar=c(1,1,1,1), pty = "s")

# Band width=0.5
> plot(ColData, border="grey60")
> plot(nb1, coords, add=TRUE, pch=19, cex=0.5)
> text(bbox(ColData)[1,1] + 1, bbox(ColData)[2,2], labels="0.5")
> box()

# Band width=0.8
> plot(ColData, border="grey60")
> plot(nb2, coords, add=TRUE, pch=19, cex=0.8)
> text(bbox(ColData)[1,1] + 1, bbox(ColData)[2,2], labels="0.8")
> box()

> par(mfrow=c(1,1))
```

Assignment III
o Provide the plot for distance-based neighbors in your report.
o Ask yourself how to interpret the plot. (You do not need to report your interpretation)

4. Global spatial autocorrelation

Global spatial autocorrelation is a measure of the overall clustering of the data. One of the most widely used statistics measuring global spatial autocorrelation is Moran's I. We will test the spatial autocorrelation of the crime data using Moran's I. Note: The `CRIME` data are ratio data since they represent residential burglaries and vehicle thefts per thousand households in the neighborhood.

First, we need to generate a weight list using `nb2listw()`. Then we use `moran()` to calculate Moran's I. The Moran scatterplot is generated by calling `moran.plot()`.

```
> col.W = nb2listw(col_nb, style="W")      # weight list
> crime = ColData$CRIME
> col_I = moran(crime, col.W, length(col_nb), Szero(col.W))
> col_I$I     # get Moran's I
> dev.off()
> moran.plot(crime, col.W, pch=19)   # Moran scatterplot
```

Assignment IV
o Insert the Moran's I scatterplot in your report.
o Based on the Moran's I value, does the crime rate show spatial autocorrelation?

Furthermore, we can perform a Moran's I test or Geary's C test for spatial autocorrelation in the `CRIME` variable (assuming normality):

```
> moran.test(crime, col.W,zero.policy=T)
> geary.test(crime, col.W,zero.policy=T)
```

The p-value reported in the test above is for the one-sided hypothesis that values occur at random throughout the spatial domain. A low p-value suggests that the `CRIME` variable exhibits significant "clustering" or positive spatial dependence.

5. Local spatial autocorrelation

Local indicators of spatial association (LISA) are statistics used to identify local clusters in the spatial arrangement of lattice data. There are two types of local spatial indicators: Anselin's local Moran's I and Getis-Ord's G_i and G_i*.

(1) Anselin's Local Moran's I

First, calculate local Moran's I using `localmoran()`.

```
> locI = localmoran(crime, col.W, zero.policy=T)
> s = dim(locI)[1]
> locI = cbind(locI,matrix(0,s,1))
```

Second, identify clusters for Local Moran's I.

```
> idx1 = locI[,4]>=1.96       # index for High-high or Low-Low
> idx2 = locI[,4]<=-1.96      # index for High-Low or Low-High
> idxh = crime>=mean(crime)   # high CRIME
> idxl = crime<mean(crime)    # low CRIME
> locI[idx1&idxh,6] = 1       # high-high cluster
> locI[idx1&idxl,6] = 2       # low-low cluster
> locI[idx2&idxh,6] = 3       # high-low outlier
> locI[idx2&idxl,6] = 4       # low-high outlier

> ColData$locI = locI[,6]
```

Third, plot LISA cluster map

```
> dev.off()
> cols.c = c("ghostwhite","red","blue","lightblue","pink")
> clr.c = cols.c[ColData$locI+1]
> leglabs = c("Not Significant","High-High","Low-Low","High-Low","Low-
High")
> plot(ColData,fill=T,col=clr.c)
```

```
> legend("topleft", fill = cols.c,legend = leglabs,bty = "n", cex = 0.8)
> title(main = "LISA Cluster Map(CRIME)")
> box()
```

Assignment V
o Insert the LISA cluster map in your report and briefly interpret it.

(2) Getis-Ord's G_i and G_i**

There is a slight difference between G_i and G_i*. The weight matrix used to calculate G_i does not include the neighbor link to the element itself, while G_i* includes self-weight (W_{ii} >0). High positive values indicate the possibility of a local cluster of high values of the variable being analyzed; very low relative values a similar cluster of low values.

- Calculate G_i

```
> col.W = nb2listw(col_nb, style="W") # same as we used for moran() and localmoran()
> G = localG(crime, col.W)     # Gi
> Gc = matrix(0,s,1)
> Gc[G>=1.96] = 1     # index for HH
> Gc[G<=-1.96] = 2    # index for LL
> ColData$G = Gc
```

- Calculate G_i*

```
> col.W2 = nb2listw(include.self(col_nb), style="W")
> G2 = localG(crime, col.W2)    # Gi*
> Gc2 = matrix(0,s,1)
> Gc2[G2>=1.96] = 1     # index for HH
> Gc2[G2<=-1.96] = 2    # index for LL
> ColData$G2 = Gc2
```

Then plot the cluster map.

```
> dev.new(width=8, height=4)
> par(mfrow = c(1, 2), mar=c(1,1,1.5,1), pty = "s")
```

- Plot G_i

```
> cols.c = c("ghostwhite","red","blue")
> clr.c = cols.c[ColData$G+1]
> leglabs = c("Not Significant","High-High","Low-Low")
> plot(ColData,fill=T,col=clr.c)
> legend("topleft", fill = cols.c,legend = leglabs,bty = "n", cex = 0.8)
> title(main=expression(paste("Values of the ", G[i], " statistic")))
> box()
```

- Plot G_i*

```
> cols.c = c("ghostwhite","red","blue")
> clr.c2 = cols.c[ColData$G2+1]
> leglabs = c("Not Significant","High-High","Low-Low")
> plot(ColData,fill=T,col=clr.c2)
> legend("topleft", fill = cols.c,legend = leglabs,bty = "n", cex = 0.8)
> title(main=expression(paste("Values of the ", G[i]^"*", " statistic")))
> box()
> par(mfrow = c(1, 1))
```

Assignment VI

- Insert the G_i and G_i* cluster maps in your report, briefly interpret it, and compare them with the LISA cluster map.

Part II: Spatial Regression Models Using R

6. Residual dependence test

Recall that we did a simple non-spatial regression in the exercise of Lab 1. It is a regression of Columbus crime with covariates (housing value [HOVAL] and household income [INC]):

```
> Col.lm=lm(CRIME ~ INC + HOVAL, data=ColData)
> summary(Col.lm)
```

If our goal is to model Columbus crime in terms of the covariates using OLS, then an assumption of independent residuals must be warranted. We can get the residuals and create a residual map.

```
> res=resid(Col.lm)
> res=(res-min(res))/diff(range(res))     # standardize
> plot(ColData,forcefill=FALSE,col=gray(1-res))
```

One way to formally test for independence in the residuals from such a regression model is to use the Moran's I test for residuals (note, this test statistic is algebraically different from the test statistic used for CRIME itself):

```
> lm.morantest(Col.lm,col.W)
```

Assignment VII

- Provide the residual plot. Does the map of residuals indicate spatial dependence (and thus suggest dependent errors)?
- Report the results of the Moran's I test. Can we accept the hypothesis of independence in the residuals?
- Compare the Moran's I and p-value with that for the CRIME variable itself (Section 4 above) and explain the difference.

7. Spatial Proximity Matrices

We are interested in explaining crime in Columbus in terms of a set of covariates (e.g., income level and housing value). If the residuals resulting from a linear model regressing crime on the covariates meet the assumption of independence, then an OLS model with independent errors might be a reasonable model to use. However, the test for independence in the residuals (Moran's I) in Section 6 revealed that the errors are indeed dependent. Thus, if we want to regress crime on the covariates of interest, we need to account for the dependent errors. Also, because the data are lattice data, we should formulate the model in an SAR or CAR specification.

In order to implement an SAR or CAR model, we need to define our spatial proximity matrix. We, in fact, did this for the Moran's I tests as well, although we only mentioned it briefly. First, let us explicitly define the proximity matrices (in terms of a list):

```
> col_nb = poly2nb(ColData, queen = TRUE)   # recall that we have done this in Part I, 3(1)
> col.listw=nb2listw(col_nb,style="W")
```

The command above uses the neighbor information in `col_nb` to define a proximity matrix for these data, where the rows of the matrix are normalized to sum to one. This matrix will work fine for Moran's I tests and for SAR models, but recall that we will need to ensure the proximity matrix is symmetric for use in fitting CAR models. One way to do this is to use an algorithm that forces it to be symmetric:

```
> col.listw.sym=similar.listw(col.listw)
```

The proximities in this case are specified in a list format; if you want the actual matrices, you could use the following commands:

```
> W=as.matrix(as_dgRMatrix_listw(col.listw))
> W.sym=as.matrix(as_dgRMatrix_listw(col.listw.sym))
```

If you would like to view these elements as an image plot, use the following command:

```
> image(W) #gives image plot of sparse matrix
             #(non-zero elements are lighter in color and zeros are white)
> image(W.sym)
```

8. Fitting SAR and CAR models

First, fit the SAR model for crime with mean given by a linear combination of the 'INC' and 'HOVAL' variables:

```
> crime.sar=spautolm(CRIME~INC+HOVAL,data= ColData,col.listw,family="SAR")
> summary(crime.sar)
```

Note that the summary function returns the parameter estimates, standard deviations, and significance levels, in addition to an estimate of the spatial autocorrelation parameter (denoted Lambda in the R output, but called "rho" in lectures) and its significance. Note that a function called `errorsarlm` also fits SAR models. The difference is that the function used above (`spautolm`) also fits CAR models.

Now, fit the CAR model for crime using the same covariates, but instead using the symmetric proximity matrix:

```
> crime.car=spautolm(CRIME~INC+HOVAL,data= ColData,
                     col.listw.sym,family="CAR")
> summary(crime.car)
```

Feel free to experiment with different specifications of the spatial proximity matrix (see the help on nb2listw for additional details). Note that parameter estimates can be quite different depending on the chosen proximity matrix. This is not surprising, as there is no definitive choice for the weight matrix; that choice is up to you, the modeler. At the very least, when using these models, one must be clear about the particular specification of proximity matrix.

Assignment VIII
o Report the fitted results for SAR and CAR model and compare the results.

Part III: Lattice Data Analysis using Geoda (Practice ONLY)

Note: Please do NOT turn in the results in this part. This part provides some useful practices if you want to use Geoda in the final project.

The following steps are based on GeoDa version 1.6.7. If you are using other versions, the interface might be a little different. However, the function won't be much different.

1. Start GeoDa and set up data

Columbus crime dataset from Lab 1 will be used for the exercise of modeling spatial autocorrelation in GeoDa. Launch GeoDa. File → New Project from → ESRI Shapefile (*.shp) (Figure 1). Select columbus.shp. For the meaning of the attribute table, open columbus.html.

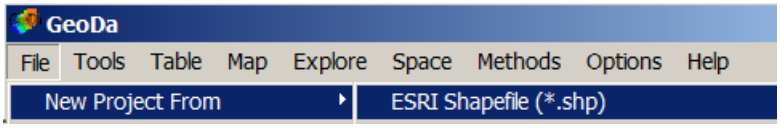

Figure 1 Create New Project

2. Create spatial weight (or spatial link) file

GeoDa provides functions to create spatial weights. The spatial weight can be generated in a contiguity method (.GAL) or distance method (.GWT) as shown in Figure 2. We will use a contiguity of polygons to generate spatial weight with the Queen option.

- Launch the popup window for creating a spatial weight by clicking Tools -> Weights ->Create. Note: Before you create weight, you need to open columbus.shp first if you haven't done so.
- Specify options as Figure 2.
  - Weights File ID Variable: **POLYID**
  - **Queen** contiguity
- Then, click **Create** button

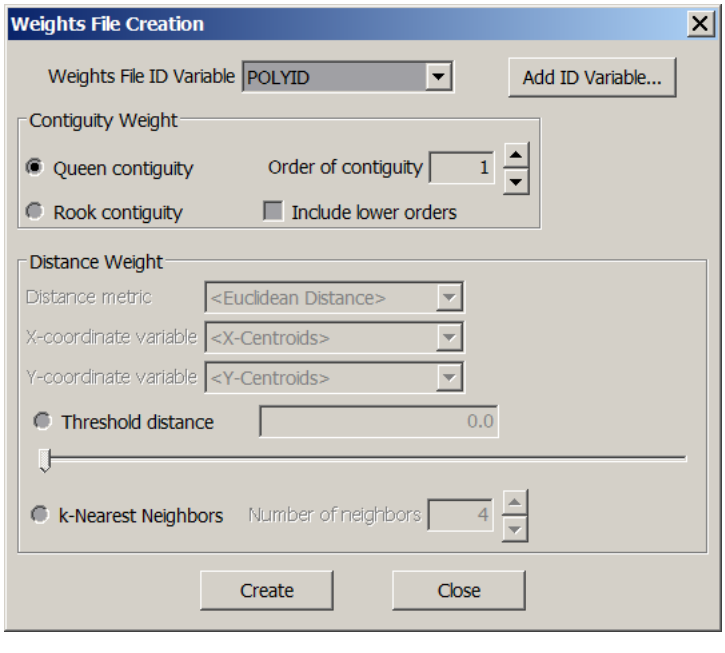

Figure 2 Interface to create spatial weight in GeoDa

In the pop-up window, save the weight file as **columbus.gal** (Figure 3). After save the file, close the "Weight File Creation" window.

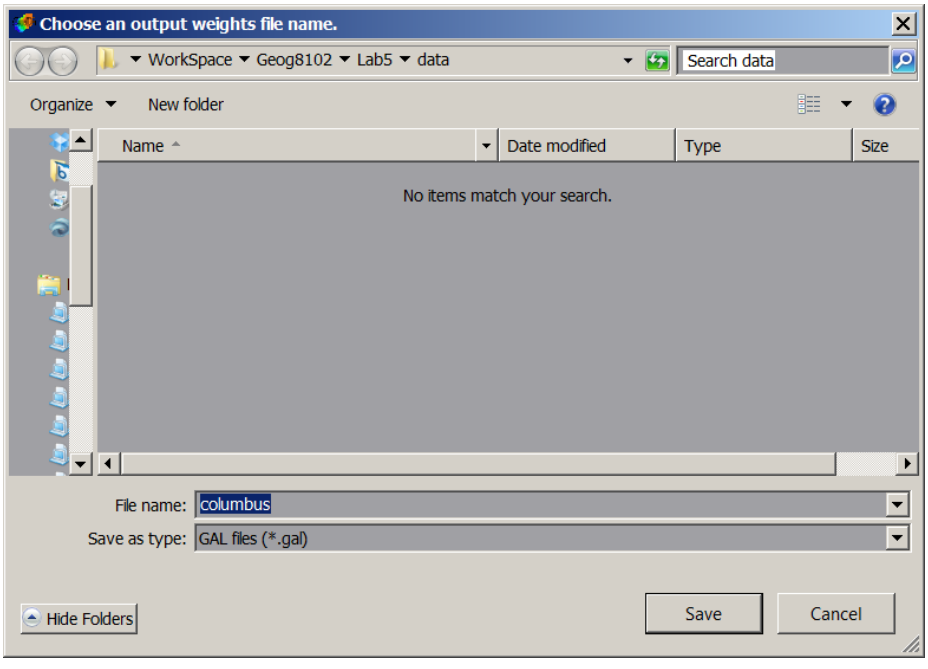

Figure 3 Save weight file

In order to get information of the spatial weight,

- Choose Tools -> Weights -> Connectivity Histogram.

Assignment B-I
○ Insert the graphical output for the connectivity of columbus.shp file. Compare it with the results we get using R (**Assignment I**). Are they the same?

3. Measure Moran's I for crime rate in Columbus

In order to evaluate if the crime rate is spatially autocorrelated or not, let us calculate Moran's I value for the variable.

- Choose Space -> Univariate Moran's I.
- Select **CRIME** variable on the popup window.
- Select **columbus.gal** file for spatial weight if there is a popup window ask you to select weight file.
- Click **OK** button.

Assignment B-II
○ Insert the Moran's I scatterplot in your report. ○ Make a choropleth map for the CRIME variable using the quantile option with 6 classes. Include the choropleth map and explain the spatial pattern.

4. Test for spatial autocorrelation using permutation methods

GeoDa provides a permutation method to conduct a statistical significant test of spatial autocorrelation. To conduct the permutation test, you need to have Moran's I scatterplot first.

- If necessary, create Moran's I scatterplot for CRIME variable as above (Section 3).
- Right click on the Moran's I scatterplot, choose Randomization -> 999 Permutations as Figure 4.

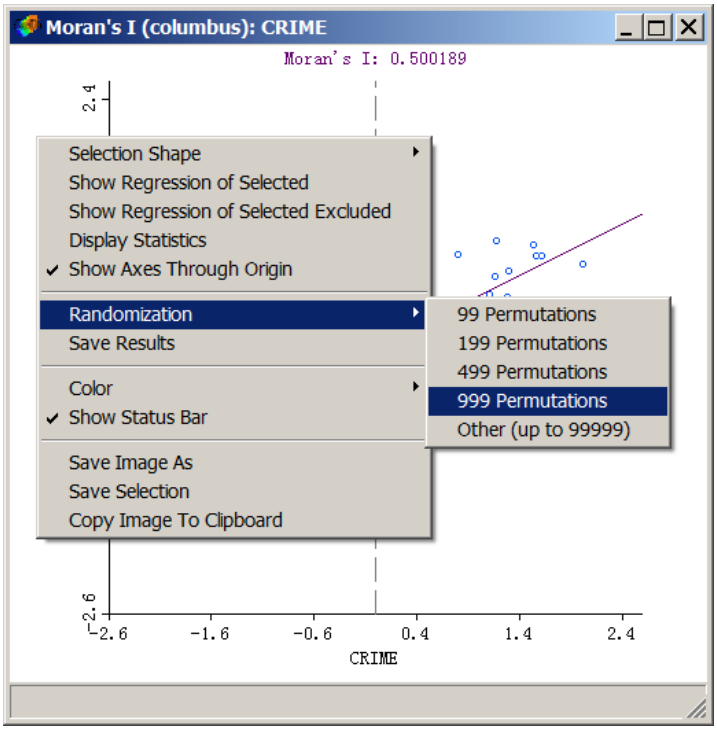

Figure 4 Interface of Permutation test for spatial autocorrelation

Assignment B-III
○ Insert the result of the permutation test for Moran's I. ○ Please interpret the result. ○ At $\alpha = 0.05$, are the CRIME rate data spatially autocorrelated?

5. Test for spatial autocorrelation in residuals of OLS regression

Run regression model with **CRIME** as dependent variable and **HOVAL** & **INC** as independent variables. To test if there is spatial autocorrelation in the residuals, please save the residuals (Figure 5).

- To run a regression model, choose the menu Methods -> Regress
- Specify **columbus.gal** for spatial weight
- Choose **Classic** regression
- Click **Run** button.

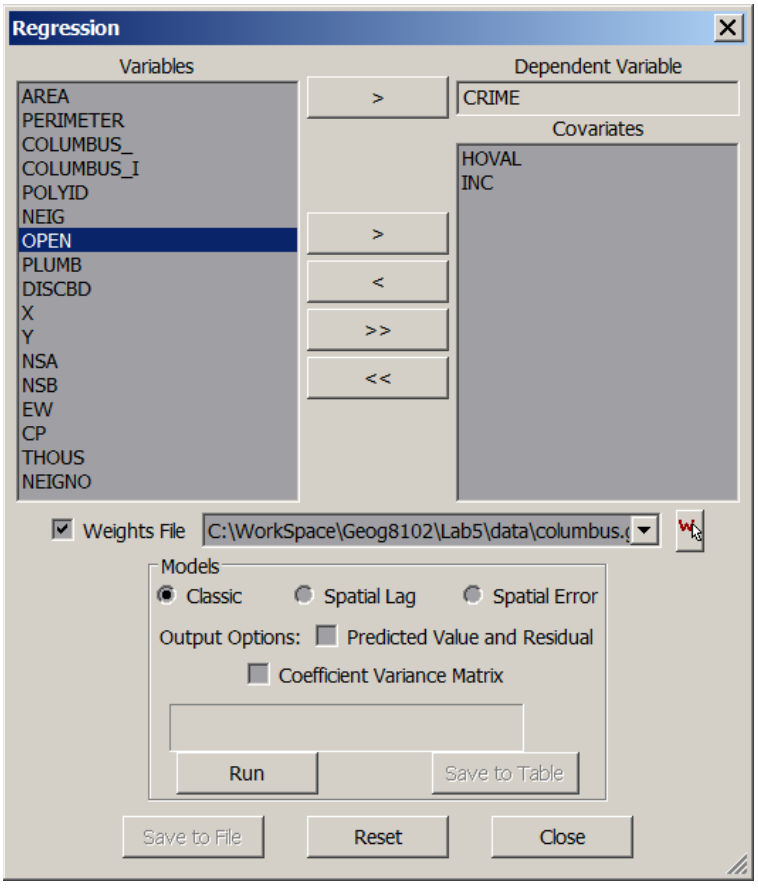

Figure 5 Regression

- Click Save to File (Figure 5), and name it as **Col_OLS.txt**. Your results have been saved in the txt file.

- Click **Save to Table** button on Regression window to save the results to the table. On the pop-up window (Figure 6):
    - Check the boxes of **Predicted Value** and **Add variable**, and keep the default in the pop-up window, and click **Add**.
    - Check the boxes of **Residual** and **Add variable**, and keep the default in the pop-up window, and click **Add**.
    - Press OK. It is temporarily added to the table of shape file.

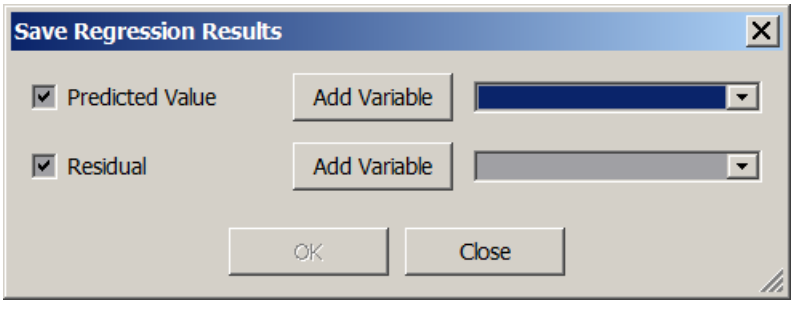

Figure 6 Interface for saving regression results as variables in the data table

Assignment B-IV

- Provide the regression results including regression summary and diagnostics for spatial dependence.
- Provide the equation for the linear regression model.
- Based on Moran's I under diagnostics for spatial dependence, is there spatial autocorrelation in the residuals?
- For the residuals, conduct a permutation test for spatial autocorrelation (please refer to Figure 3). Report the result. Based on the permutation, is there spatial autocorrelation in the residuals?
- Make a choropleth map for the residuals using the quantile option with 6 classes. Include the choropleth map and compare it with the choropleth map for the CRIME variable.

6. Run a spatial lag regression model and test for spatial autocorrelation for its residuals

Run a spatial lag regression model with the same variables as above.

- Refer to the above steps, run a spatial lag regression model. Only difference is to choose **Spatial Lag** instead of Classic.

Assignment B-V

- Provide the spatial lag regression results including regression summary and diagnostics for spatial dependence (note: spatial dependence test is not on the residuals of spatial lag model. It is test to compare the classical OLS model to alternative spatial lag model).
- Provide the equation for the spatial lag regression model.
- For the residuals of the spatial lag regression, conduct a permutation test for spatial autocorrelation (please refer to Figure 4). Report the result. Based on the permutation, is there spatial autocorrelation in the residuals?
- Make a choropleth map for the residuals of spatial lag regression using the quantile option with 6 classes. Include the choropleth map and compare it with the choropleth maps of the OLS residual map.

7. Run a spatial error regression model and test for spatial autocorrelation for its residuals

Run a spatial error regression model with the same variables as above.

- Refer to the above steps, run a spatial error regression model. Only difference is to choose **Spatial Error** instead of Classic.

Assignment B-VI

- Provide the spatial error regression results including regression summary and diagnostics for spatial dependence.
- Provide the equation for the spatial error regression model.
- For the residuals of the spatial error regression, conduct a permutation test for spatial autocorrelation (please refer to Figure 4). Report the result. Based on the permutation, is there spatial autocorrelation in the residuals?
- Make a choropleth map for the residuals of spatial error regression using the quantile option with 6 classes. Include the choropleth map and compare it with the residual maps of OLS and spatial lag model.

-End-