

ASSIGNMENT 7 (Due Monday, May 9)

This final study is a continuation of the analysis in Assignment 6, and will use the results of that analysis. It is assumed that you have constructed the shapefile, **Final_Tracts.shp**, in ARCMAP, and also the MATLAB workspace **Phil_Housing.mat**. Here you will carry out spatial regressions that take account of the *spatial autocorrelation* found in the OLS residuals when regressing the log of Median Housing Value (**lnMV**) on Percent Housing Vacancy (**%Vac**) and Percent NonWhite (**%NW**). First open ARCMAP, and create a blank map document. Now add **Final_Tracts.shp** to this document and, for convenience, re-label this layer as **Philadelphia Housing**. Save the new document as **Phil_Housing_7** (for Assignment 7).

- (a) In this initial phase of the study you will construct a new spatial weight matrix for Philadelphia based on *boundary shares*. This type of weight matrix is more appropriate for areal data such as contiguous census tracts. The procedure for doing so is outlined in the CLASS NOTEBOOK on the class web page. Scroll down to part IV and open section **3.2.3 in Using Matlab**, called “Making Boundary-Share Weight Matrices using ARCMAP and MATLAB”. Here a detailed procedure is outlined using the **Eire** data set illustrated in class.
1. In using this procedure you can here start at step (1.2) by constructing a second copy of the shapefile, **Final_Tracts**, say **Final_Tracts_0**. Now continue the procedure up to Step (1.5).
 2. In Step (1.5) you will obtain a file, **Final_Tracts_Intersect**. (This may take several minutes to complete.) If you open the Attribute Table as in step (1.6) and look at the bottom of the window, you should see that there are 3507 polyline features, representing shared boundary segments. Working with feature files this size can be very slow – so be patient!
 3. When the new field, **HowLong**, is fully calculated this will give you the lengths of all shared boundaries. To gain some feeling for how this works, select the column, **ID1**, just to the left of **HowLong**, that contains the row IDs of one of the tracts sharing the boundary segment in that row. Reorder this column in ascending order (right click on the column and select **Sort Ascending**) and now scroll down to **ID1 = 284**. You should see seven rows with 284. Select the first row (**HowLong = 9471.09**), find the selected segment on the map, and enlarge that area. You should now see that this census tract (**FID = 283**) appears to be a “peninsula” defined by two missing tracts (with zero population) connected by the Schuylkill River. Notice in particular that this boundary-share procedure excludes connections between this census tract and all others on the east side of the river. However, if this map had been drawn more crudely (or if the river had been very narrow) so that the river was a *single curve*, then tracts on the east side would now share boundaries with this tract. Since these conventions are often arbitrary, you should be aware that boundary shares can sometimes yield strange results. Hence you should always *look at the map* to be sure that these results are reasonable. Note also that the remaining six rows come in pairs, and refer to the segments shared with three

other tracts (FID = 268,272,281). So boundaries shared with no other tract appear once (with **ID = ID1**) and those shared with other tracts appear twice. [This distinction is used in the MATLAB program, **shared_bd_lengths.m**, below to identify those boundary segments shared with other tracts.]

4. Now carry out the steps up to (2.1) where you should obtain a text file, **shared_bd_length.txt**, containing only columns, (**ID, ID1, HowLong**). In step (2.2) when you import these results to MATLAB, you should open the workspace **Phil_Housing.mat** created in Assignment 6, and import the text file, **shared_bd_length.txt**, to this workspace.
5. Rename this as **DAT**, and construct **M** as in step (2.2) using the program, **shared_bd_lengths.m**. As a consistency check, now type the command

```
» M(284,:)'
```

and you should see the output

```
(269,1)    1884.6  
(273,1)     627.8  
(282,1)    2032.3
```

listing the lengths of boundary segments shared with tract 284 (FID = 283). [The reason for this simple output format is that the matrix **M** is in *sparse* form (as can be seen by both the matrix icon and class label in the workspace). Note also the transpose at the end, which converts rows to columns.]

6. Finally, you should row normalize this matrix with the command

```
» W0 = row_norm(M);
```

This will avoid overwriting the four-nearest-neighbors weight matrix, **W**, constructed in Assignment 6. Check the workspace to be sure that **W0** is a 353x353 matrix.

Note: For *boundary shares* weight matrices such as **W0** this *row normalization* has a clearer interpretation than in most cases. Here one measures the “influence” of tract **j** on tract **i** in terms of the fraction of total shared-boundary for tract **i** that is shared with tract **j**. Notice also that unlike most spatial weights, this measure is fundamentally *asymmetric*. In particular if total shared boundary of tract **j** is relatively large compared to that of tract **i**, then the “influence” of **i** on **j** may well be smaller than that of **j** on **i**.

7. Since **W0** is also seen to be in sparse form. we can now represent row 284 of **W0** by writing:

» **W0(284,:)** '

to obtain the output:

(269,1)	0.41468
(273,1)	0.13814
(282,1)	0.44718

This is seen to be a normalization of the lengths above, which now add to one. [It is worth noting here that if a given matrix, **A**, is not sparse, you can make a *sparse copy*, **AA**, by writing

» **AA = sparse(A);**

This will allow you to examine the rows (or columns) of **A** in the above format by using **AA**.]

8. Finally, as a second illustration that these weights need not always be the most “reasonable” ones, write:

» **W0(353,:)** '

Check this result in ARCMAP and comment on your findings.

- (b) Given this boundary-share weight matrix, **W0**, it is now appropriate to recheck for the presence of spatial autocorrelation in the OLS residuals by using **W0**. Rather than regressing on nearest-neighbor residuals, you will here use the more flexible approach based on the *random permutation tests* in the MATLAB program **sac_perm**. This program allows *any* weight matrix to be used, and also reports three different indices (**I**, **rho**, **r**). The first, (Moran’s) **I**, is the most commonly used measure of spatial autocorrelation. The second, **rho**, is very similar to the JMP regression of residuals, **res**, on their weighted neighbors, **W0*res**, and gives essentially the same result.

1. In the workspace you should already have the vector of OLS residuals, **res**, constructed in Assignment 6.
2. To perform the random-permutation test with 999 random permutations, write:

» **sac_perm(res,W0,999);**

You will only need the Screen output, and not the Data output.

3. Given this result, comment on the presence of spatial autocorrelation in these OLS residuals. Note in particular how the values of these indices related to the interval of simulated values represented by the MAX and MIN values in the screen output.

(c) The first spatial regression you will perform will use the **spatial autoregression (SAR) model**, in the MATLAB program, **sar.m**. To do so, open this program (» **edit sar**) and review its input requirements.

1. To construct the regression data, recall from JMP that data for the dependent variable, **lnMV**, is in the 4th column of **Phil_matrix**, and data for the explanatory variables, **%Vac** and **%NW**, is in the 5th and 6th columns, respectively. So construct the data inputs, **y** and **X**, by writing:

```
» y = Phil_matrix(:,4);  
» X = Phil_matrix(:,5:6);
```

Before proceeding, display the first few rows of **y** and **X** and compare with JMP to be sure that you have selected the right columns. (If you mistakenly use **MV** rather than **lnMV** your results will be *very* badly behaved.)

2. To make the list of variable names, **vnames**, write:

```
» vnames = strvcat('%Vac','%NW')
```

By leaving off the semicolon, the vector of names should now be displayed on the screen.

3. Given the size of the weight matrix, **W0**, it is worthwhile calculating the eigenvalues of **W0** before hand, and using these as inputs to the program. To do so, write:

```
» eigvals = real(eig(full(W0)));
```

Here the function **full** converts the (default) sparse representation of **W0** into a full 353-square matrix, and the function **eig** then computes its eigenvalues. Finally, **real** removes any (small) imaginary components that may have crept in during the root extraction process.

4. To run the **spatial autoregression** write:

```
» [OUT,cov,DAT] = sar(y,X,W0,vnames,eigvals);
```

Copy-and-paste the screen output to a WORD file for later comparison with the spatial-lag model.

5. Next, noting from the list of **sar.m** outputs that the SAR-residuals and OLS-residuals are in the 3rd and 5th columns of **OUT**, respectively, write:

```
» res_OLS = OUT(:,5);  
» res_SAR = OUT(:,3);
```

Observe that **res_OLS** should be essentially the same as **res** (from JMP). A simple way to verify this is to write:

```
» max(abs(res_OLS – res))
```

This will display on the screen the maximum absolute difference between the components of these two residual vectors. [This will not be zero, but should be very small (around 10^{-6}). If not, check to be sure you have defined **res_OLS** correctly.]

6. Next, use **sac_perm** on the SAR residuals:

```
» sac_perm(res_SAR,W0,999);
```

A comparison between these significance values and those obtained above for the OLS residuals should give you a clear idea as to whether SAR has effectively removed the spatial autocorrelation present in the simple OLS analysis.

7. Comment on your findings. How does the significance test for the model parameter, **rho** (in the “AUTOCORRELATION RESULTS”), support these findings?

(d) Next, you will carry out a second spatial regression using the **spatial lag (SL) model**, using the MATLAB program, **sp_lag.m**. As before, open this program (» **edit sp_lag**) and review its input requirements.

1. The data inputs, **y** and **X**, are the same for this model. Also the weight matrix, **W0**, variable names vector, **vnames**, and eigenvalue vector, **eigvals**, are the same.

2. So to run this program, write:

```
» [OUT,cov] = sp_lag(y,X,W0,vnames,eigvals);
```

Copy-paste the screen output to the WORD file created above, so that the two sets of outputs can be compared side-by-side.

3. Noting from the list of **sp_lag.m** outputs that the SL-residuals are again in the 3rd column of OUT, save these as:

```
» res_SL = OUT(:,3);
```

4. As before, use **sac_perm** on these SL residuals:

```
» sac_perm(res_SL,W0,999);
```

Compare these significance values with those obtained above for the SAR

residuals, and comment on your findings. In particular, state whether or not these results appear to be consistent with the test results for the **Common Factor Hypothesis** above.

5. How does the significance test for the model parameter, **lam** (in the “AUTOCORRELATION RESULTS”), compare with that for **rho** in the SAR model?
- (e) Given these two sets of regression results, compare the “GOODNESS-OF-FIT” measures for each. Focus in particular on the **Pseudo R-Square value**, which is in many ways the most meaningful of these.
- (f) You will now compare the residuals for these models **spatially** to see whether the relative **patterns** of spatial residuals agree with the statistical results above. To do so:

1. First combine the three sets of residuals into a common matrix:

» **RES = [res_OLS,res_SAR,res_SL];**

2. Next, to allow this data to be merged with ARCMAP data, add a copy of the appropriate **FID** column by writing:

» **RES(:,4) = [0:352] ' ;**

Remember that **FID** starts with ‘0’ rather than ‘1’.

3. Now, export this file to your home directory with a command similar to:

» **save 'e:\Home\Residuals.txt' RES -ascii**

4. As usual, open this file in EXCEL, choose an appropriate numerical format [as in part 10.(a) of Assignment 5] and add a title row with column labels (**res_OLS,res_SAR,res_SL,ID**).

5. Save as a **tab-delimited** text file, and rename this file as **Residuals.tab**.

6. Now open your map document, **Phil_Housing.mxd**, in ARCMAP, and add **Residual.tab** to the data frame “Philadelphia Tract Data (1990)”.

7. Merge this data with the attribute file for “Median Housing Values” (**Joins and Relates → Join**) using **FID** and the constructed column **ID** as the common attributes for the merge.

8. Save as a new shape file (**Data → Export Data**) named **Resid_Data.shp** and add to the Table of Contents.

9. Rename the layer as “OLS Residuals”, and display these as follows:
 - (i) First, select **res_OLS** as the variable in the **Symbology** window.
 - (ii) Now, open the **Classify** window and set the **Classification Method** = “Standard Deviation”. [This is the best way to make all sets of residuals comparable.] Click **OK** and return to the **Symbology** window.
 - (iii) Set the colors manually, using “White” for the **mid range** (-0.50 – 0.50 Std Dev). Use increasing shades of “Blue” for the **negative** ranges and “Red” for the **positive** ranges. [These colors are chosen in order to facilitate visual comparison of the three sets of residuals. However, if you plan to print these in **black-and-white**, then it would be better to use a single gray scale, say running from light to dark as values increase.]

 10. Next, use **Copy** and **Paste Layer** to add two more copies of the “OLS Residuals” layer.
 - (i) Name them as the “SAR Residuals” and “SL Residuals”.
 - (ii) Repeat step 9 for each of these layers, using the respective variables, **res_SAR** and **res_SL**. (Don’t worry if the number of “Classes” differs for each. Classes are assigned automatically according to how many Standard Deviation classes are present in the data distribution.)

 11. Using **Edit → Copy Map to Clipboard**, copy all three sets of residuals into WORD, with **OLS Residuals** in the middle, and compare them visually. Do these different patterns agree or disagree with your statistical findings? Be explicit.
- (g) Finally, consider the **regression results** in all three cases.
1. How do the **beta coefficients** compare between the models? Are there any notable differences in sign or magnitude? If so, how might you account for this?
 2. What about differences in the significance levels (**P-values**) of the coefficients? Again, how might you account for this?
 3. If you had to pick one of these models as “best”, which would you choose and why?