3.2 TIPS FOR USING MATLAB

3.2.1 Exporting Graphics from MATLAB to WORD and Editing Details

Given a graphical plot in MATLAB, say **graph.fig** , the following procedure allows this graph to be imported into WORD and edited in a very flexible way:

1. Export **graph.fig** from MATLAB as an Enhanced Meta File (**.emf**).

2. Insert **graph.emf** into WORD as a picture (from file).

    a. In WORD click anywhere on the Figure and use the mouse to resize the Figure as desired. (This must be done first in order to preserve sizes between the Figure and all its labels.)

    b. Next click **Format** → **Picture** (or **Object**) and under **Layout** set **Wrapping style** = "Behind text".

    c. Now you can **drag** the Figure to any position on the page.

3. To edit this Figure in WORD:

    a. Right click anywhere in the Figure and click: **Grouping** → **Ungroup** (if asked to convert Drawing Object to Picture, say YES).

    b. Then left click anywhere outside the Figure to hide all boxes.

4. You can now edit any **line object** or **text object** in the Figure by using the standard drawing tools in WORD. Be sure that the Drawing toolbar is active (**View** → **Toolbars** → **Drawing**)

    a. For example, you can select a line (or curve) and change its **style** and **weight**.

    b. Also you can select a word and change its **font** or **size**.

5. To create a **vertical label** for a graph:

    a. Click on the label so that the text box appears, and now click the border of the text box so that the border is **dots** rather that **slashes**.

    b. On the main toolbar click: **Format** → **Text Direction**

   c. Set the text to **vertical**, and change the shape of the text box to accommodate the vertical text.

6. When finished editing the Figure, be sure to **regroup** the Figure:

   a. On the Drawing toolbar click on the **Select Objects** arrow, and use the mouse to select all relevant parts of the Figure, including all labels.

   b. Again right click anywhere in the Figure and click: **Grouping** → **Group**.

   c. You should now be able to move the whole Figure (plus labels) anywhere on the page.

7. Finally, to insert **additional discussion text** next to the Figure, say a paragraph on the left side of the Figure:

   a. Start a new paragraph (so that you can reset margins). Move the **Right Indent** marker to the left, far enough so that the Figure will fit to the right of the text.

   b. Continue typing text until you have gone beyond the bottom of the Figure.

   c. Again start a new paragraph, and reset the **Right Indent** to its original position.

8. Alternatively, you can add a two-column **Table** and place the Figure in the right hand column. To do so:

   a. Start at the point in the text where you want the [text + Figure] to be.

   b. Click: **Table** → **Insert** → **Table** , and make a Table with two columns and one row.

   c. Use the mouse to move the vertical centerline of the Table so that the width of the right-hand cell will accommodate the Figure.

   d. Start typing text from the top of the left cell, using a manual return to avoid the vertical centerline.

   e. Keep typing until the vertical height of the Table will accommodate the Figure.

   f. Now drag the Figure into the right-hand cell.

g. Finally, to be sure that the Table has an invisible frame, put the mouse anywhere inside the Table, and click on the box that appears in the upper left-hand corner of the Table. This will select the entire Table.

h. With the Table selected, click: **Table → Table Properties → Borders and Shading** .

i. Under "Border Settings" click **None**. This will remove any border lines from the printed form of the Table.

3.2.2 Making Boundary-Share Weight Matrices in MATLAB

The following procedure is useful if one has a boundary file in MATLAB format and wants to construct a boundary-share weight matrix without exporting the file back to ARCMAP (using BND2SHP.exe). Here we assume that one has a matrix, **boundary**, in the MATLAB workspace that is in standard boundary format. [This can be checked quickly with the command >> **Z = polyform(boundary)**. If there are no error messages, then the file is in correct format, and the number of rows in **Z** will tell you how many polygons are in your boundary file.]

(1) To calculate boundary shares for this file use the program **bnd_shares_matrix.m** found in the class MATLAB directory. For the file **boundary**, the appropriate MATLAB command is:

```
» W = bnd_share_matrix(boundary);
```

(2) The output **W** is a **row-normalized weight matrix** of boundary shares, which can be used directly in any spatial autoregressive model.

3.2.3 Making Boundary-Share Weight Matrices using ARCMAP and MATLAB

The following notes outline a procedure for identifying lengths of shared boundaries in ARCMAP 10.6. These are then imported to MATLAB and boundary-share weight matrix will be constructed. For concreteness, we will use the shapefile, **eire_bd.shp**, in the directory **arcmap\projects\eire**.

1. Creating Shared Boundary Lengths in ARCMAP

The following procedure[1] starts by using the intersection tool in ArcMap to create shared boundaries as line features.

[1] I am indebted to Dana Tomlin for suggesting this procedure to me

(1.1) First copy **eire_bd.shp**, to you home directory. Now open a new map file in ARCMAP, import this copy of **eire_bd.shp** and save it to your home directory as say **eire_boundary.mxd**.

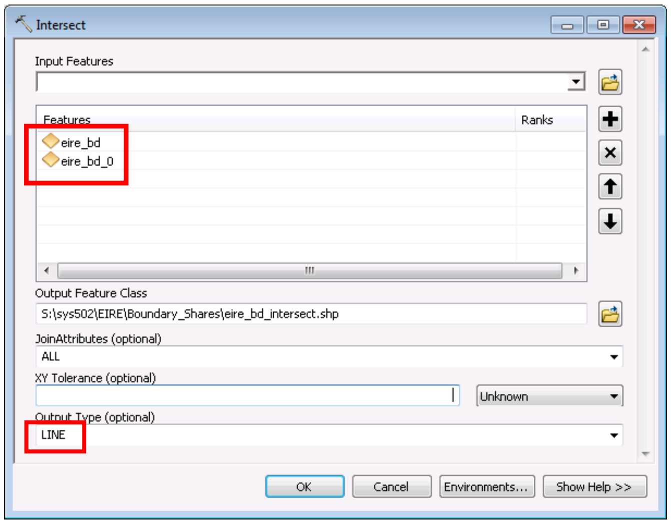
(1.2) Open the attribute table of **eire_bd.shp** and create a new field, **ID**. Calculate values as ID = [FID] + 1 (so rows are numbered starting with 1 in this column).

(1.3) Next make a second copy of **eire_bd.shp** by (i) right clicking on this layer, (ii) selecting **Data** → **Export** , and (iii) saving the copy as, say, **eire_bd_0.shp** (be sure to set **Save as type** = "Shapefile").

(1.4) Now open the intersect tool in ARCTOOLBOX:

Analysis Tools → Overlay → Intersect

The task is to intersect **eire_bd.shp** with the copy, **eire_bd_0.shp** of *itself*, and to save the result as *polyline features*. In the Intersect window, first open **Input Features** and click on "eire_bd". Then repeat the procedure for "eire_bd_0", so both appear, as shown. Now *scroll* to the bottom of the window and set **Output Type** = LINE, as shown. Be sure to specify the path (in Output Feature Class), where the new file, say **eire_bd_Intersect.shp**, is to be stored.



Output Feature Class: S:\sys502\EIRE\Boundary_Shares\eire_bd_intersect.shp

JoinAttributes (optional): ALL

Output Type (optional): LINE

(1.5) When you click **OK**, the intersect tool will run, and will produce a new shapefile called **eire_bd_Intersect** in the Table of Contents.

(1.6) If you open the Attribute Table of **eire_bd_Intersect.shp** you will see that there are now 245 polyline features. Each feature corresponds to the line segment shared by two counties in Eire. The two sets of fields shown correspond to the pair of counties sharing this segment. You will see counties repeated for those segments not shared by any other county, i.e., those on the edge of Eire.

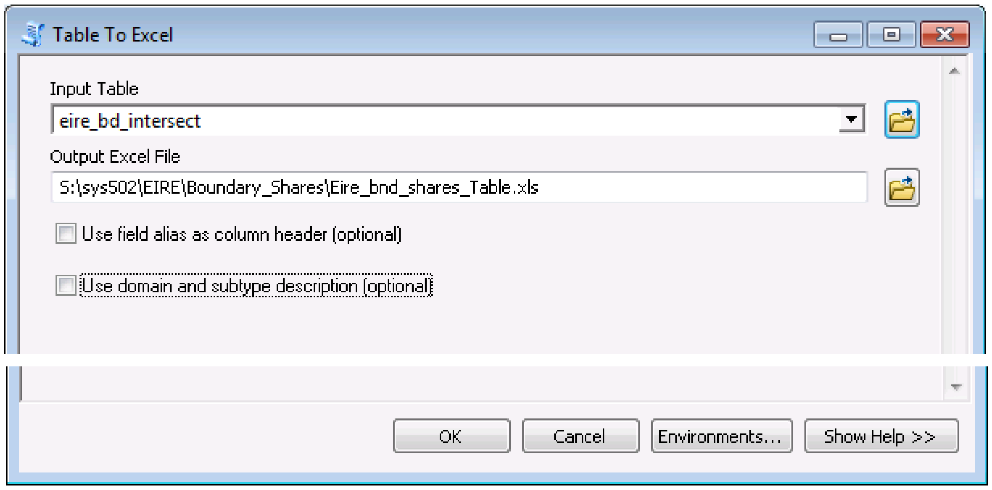
(1.7) The next task is to calculate *lengths* of each segment (even those we are not interested in). To do so, create a new field in the table (**Options → Add Field**) of type *Double* with name: **Arc_Length**. Then right click on this new column and select

"Calculate Geometry". In the top field of the window that opens set **Property** = "Length" and click **OK**. The lengths of each line segment will now appear.

2. Import to MATLAB for Calculating Shares

(2.1) To use this data in MATLAB, you must first export it to EXCEL. To do so, open ARCTOOLBOX and scroll to:

Conversion Tools > Excel > Table to Excel



Input Table
eire_bd_intersect
Output Excel File
S:\sys502\EIRE\Boundary_Shares\Eire_bnd_shares_Table.xls
Use field alias as column header (optional)
Use domain and subtype description (optional)

Click OK and this EXCEL file will appear in your chosen directory. The only columns of interest here are the two **ID** columns (one possibly labeled **ID_1**) containing the row numbers of the relevant counties for each segment, and the segment lengths, **Arc_Length**. Delete all other columns and save this file as **shared_bd_lengths.xls**.

(2.2) Now open MATLAB, set the path to the class **matlab** directory, import the file **shared_bd_lengths.xls** (as a numerical matrix), and rename it as **DAT**. If you double click on **DAT** in the workspace, it will open in the **Variables** window. By scrolling down to rows **22:25** you should see:

```
1    1    36.633
1    9    36.633
9    1    36.633
9    9    36.633
```

The first row involves only county (polygon) number 1 in Eire, and is not interesting for our purposes (similarly for the last row). The second and third rows both show the boundary length between counties **1** and **9**. This information will be used. To extract the relevant information in a usable form, you will use the program **shared_bd_lengths.m** with the command:

```
» M = shared_bd_lengths(DAT);
```

(2.3) To check the output, if you now double click on **M**, the first five entries should be:

```
(1,9)     36.633
(1,10)    79.267
(1,11)    13.211
(1,25)    55.267
(1,26)    88.45
```

To verify these results, reopen **eire_boundary.mxd** and open the Attribute Table of **eire_bd.shp**. If you select row 1 [FID = 0] you will see that this county shares boundaries with 5 other counties, namely those in rows (9,10,11,25,26) [FIDs = (8,9,10,24,25)]. You will also see that the boundary lengths shown correspond to the relative shared lengths seen on the map. So for example the length of boundary shared with county 11 [FID = 10] is the shortest, and so on.

(2.4) Finally, to make this into a row-normalized weight matrix, use the command:

```
» W = row_norm(M);
```

If you then double click on **W**, the first five entries should be:

```
(1,9)     0.13427
(1,10)    0.29054
(1,11)    0.048423
(1,25)    0.20257
(1,26)    0.3242
```

These are precisely the fractional values of the lengths shown above, indicating the boundary-share weights relevant for county 1 in spatial regression analyses.

3. Making Contiguity Matrices from Boundary Share

Finally it is important to note that contiguity weight matrices are immediately constructible from boundary shares. A contiguity matrix, **Wc** $=(w_{ij})$ is defined by

$$w_{ij}=\begin{cases}1/n_i \ , & i \text{ and } j \text{ share a boundary}\\ 0 \ , & \text{otherwise}\end{cases}$$

where n_i is the number of $j \neq i$ sharing a boundary with i. Given a boundary share weight matrix, **W**, it should be clear that a corresponding contiguity matrix, **Wc**, can be constructed by simply replacing the positive elements of **W** by ones, and then row normalizing the resulting matrix. Given a boundary share matrix, **W**, this can be quickly accomplished in MATLAB by the command:

» WW = (W > 0); Wc = row_norm(WW);

The first command is a logical operator in MATLAB that assigns a one to every position **WW**(i,j) for which **W**(i,j) > 0 is "true" and zero everywhere else.

3.2.4 Clipping Grids in ARCMAP for use in MATLAB

When conducting analyses in MATLAB such as "hot spot" analyses, it is often useful to calculate p-values for a grid of reference points over the area in question. For irregular areas such as Philadelphia, rectangular grids produce many points *outside* the boundary that are not relevant, but require much computer time for calculation. In such cases it is useful to be able to "clip" out all grid points not inside the relevant boundary. If one has a shapefile of this boundary in ARCMAP then this can be accomplished by importing the grid to ARCMAP, clipping, and re-exporting to MATLAB. Assuming the existence of an appropriate boundary file in ARCMAP, say **Boundary.shp**, this procedure can be accomplished as follows:

1. Merge Boundary File if Necessary

In order to clip grid points outside of **Boundary.shp**, it is essential that this file consist of exactly *one* polygon. If it does, then proceed to Step 2 below. But if it contains more than one polygon (say all census tracts in Philadelphia) then use the following procedure to merge these into a single polygon:

(i) First right click **Boundary.shp** and use **Data → Export Data** to make a new copy, say **Boundary_merged.shp**. Add it to the map document.

(ii) Open the Attributes Table for **Boundary_merged.shp** and select all polygons (with **Options → Select All**). Close the Attribute Table, and now all polygons should now be selected on the map.

(iii) Open the **Editor** and click **Start Editing**. In the window that opens select the source path in the top window that yields **Boundary_merged.shp** among the files in the bottom window. Click **OK**.

(iv) Now click **Editor** again and select **Merge**. In the window that opens just click **OK** (it doesn't matter which polygon you merge to). When the operation is complete, the map should now be dissolved into a single polygon. (You can check this by opening the Attribute Table of **Boundary_merged.shp** and verifying that there is now only one polygon feature).

(v) Finally, remove all selections by clicking **Selection → Clear Selected Features**. For the rest of the procedure below you will be using the file **Boundary_merged.shp** whenever it refers to **Boundary.shp**.

2. Make Reference Grid in MATLAB

To make the desired grid of reference points, you will use the program **grid_form.m** in MATLAB using the command:

```
>> G = grid_form(Xmin,Xmax,Xcell,Ymin,Ymax,Ycell)
```

To obtain appropriate parameters, open **Boundary.shp** open in ARCMAP, and use the mouse to visually select coordinates **(Xmin,Xmax,Xcell,Ymin,Ymax)** for the corners of a rectangular grid large enough to cover the whole area. Next decide on an appropriate grid spacing, usually with **Xcell = Ycell**, again by visual inspection. Given these inputs, construct the desired grid, **G**.

3. Import the Grid to ARCMAP

Next export this grid from MATLAB to the workspace as an ASCII file with the command:

```
>> save Grid.txt G –ascii
```

You will need to open this file in EXCEL to reformat for ARCMAP. In EXCEL you will see two columns of coordinates. Here it is important to transform coordinates to the same units as in ARCMAP. So for example, if you see a row value like 5.73+E5, while the appropriate value in ARCMAP should be of the form 573000 (say in feet), then you must transform these value by selecting the columns and scrolling to **Format → Cells → Number**. In this case you would choose "Decimal places" = **0**. Next, you must add "X" and "Y" labels to the columns by using **Insert → Rows**. Finally, you should save the file as a "Text (Tab Delimited)" file, and rename it as **Grid.tab** (this is the standard format for tab delimited text files in ARCMAP).

Now import the file to ARCMAP. If you get an error message, check the above steps again. You might also try using simply **Grid.txt** (which mysteriously works in some cases).

4. Clip the Grid in ARCMAP

Once the grid is in ARCMAP, right click on the file and select **Display X-Y Data**. ("X" and "Y" should already be entered for the appropriate "X Field" and "Y Field"). Click **OK**, and a new file **Grid.tab Events** should appear at the top of the Table of Contents. Use **Data → Export** to resave this as a shapefile, **Grid.shp**. Now you can clip this using the **Boundary.shp** file. To do so:

(i) Open **Arc Toolbox** and select **Analysis Tools → Extract→ Clip**.

(ii) In the window that opens, set the "Input Feature" = **Grid.shp** and the "Clip Feature" = **Boundary.shp**. The default name of the file saved will be **Grid_Clip.shp** (which can be renamed if you choose).

(iii) Click **OK**, and when the operation is complete you will see that **Grid_Clip.shp** has been added to the Table of Contents. If you close **Grid.shp** then you can see this clipped version more easily.

5. Export to MATLAB for use in Analyses

The final step of this procedure is to export the clipped grid back to MATLAB. To do so:

(i) First convert the dbf file, **Grid_Clip.dbf**, into a text file for import to MATLAB. Open EXCEL and import **Grid_Clip.dbf**. Remove any columns other that "X" and "Y".

(ii) Now save the file as a "Text (Tab Delimited)" file, **Grid_Clip.txt**.

(iii) This text file can now be directly imported into the MATLAB workspace for use in analyses.

3.2.5 Exporting Data from MATLAB to ARCMAP

To export data from MATLAB to ARCMAP, there are two procedures that can be used. The first is easiest and most efficient when data sets are not too large. The size limit relates to the maximum number of data items (524,228 cells) that can be handled by the Array Editor in MATLAB.

1. Data Set Not Too Large

If you have a data matrix with say 50,000 samples on 10 variables in MATLAB this is 500,000 cells, and is less than 524,228. So this just works.

(a) In MATLAB, double click on the matrix to be exported.

(b) In the Array Editor that opens, click **Edit → Select All** and then **Edit → Copy**

(c) Open EXCEL and click **Edit → Paste**. The data should appear exactly as in MATLAB.

(d) Now add Column Labels, following the format in ARCMAP (single strings of not more that 8 symbols, with no punctuation symbols other than "_")

(e) Save as a DBF file (simplest procedure), or Tab-Delimited TEXT file.

(f) A DBF file can be loaded directly into ARCMAP. A Tab-Delimited TEXT file must be altered further. You must change the extension .TXT to .TAB.

2. Data Set Too Large

If you cannot display the MATLAB data matrix in the Array Editor (more than 524,228 cells), then you must use the SAVE command procedure in MATLAB [as described for example in steps (6) and (7) of Class Assignment 5]

3.2.6 Converting Boundary Shapefiles to MATLAB format

In this approach we use GEODA (rather than ARCMAP) to make the transformation. Here we shall use the file, **eire_bd.shp**, for an example.

(a) We start by assuming that the boundary shapefile, **eire_bd.shp**, is open in GEODA. We also assume it has an **ID** column (easily created in ARCMAP with the calculation, **ID** = [FID] + 1).

(b) Select **Tools → Shape → To Boundary** from the Main Menu in GEODA.

(c) In the window that opens, select **eire_bd** for the input file and save as, say **eire_bd_mat.txt**. The default option for **Format Types** should be "Type 1". This is the type you want for MATLAB.

(d) Click **OK**, and the text file will be saved.

(e) If you open this text file, you will see that the first line is "**1 51**", which tells you that the first polygon is defined 51 distinct points. In order to put this in MATLAB format, the polygon must be "closed" by repeating the first point at the end of each polygon, creating a full "loop"

(f) This can be accomplished by importing the text file to MATLAB as say, **data**, and using the program, **bnd_geoda.m**, to transform it with the command:

```
>> eire_bd  =  bd_geoda(data);
```

(g) If you open this file in the **Variable Editor** of MATLAB (just double click on **eire_bd** in the workspace), you will see that the first line is now "**1 52**". This tells you that the program has successfully added an additional point to the bottom of the first polygon (and each subsequent polygon), which is a copy of the first point in that polygon.

(h) This file is now ready for use in those MATLAB programs requiring boundary inputs.