Understanding spatial weights matrices in cluster detection

Objectives:

The purpose of this exercise is to get beneath the surface of ArcGIS and to understand in greater detail how cluster detection tests work. In particular, the exercise aims to illustrate what spatial weights matrices are, how they can be calculated, and how the construction of a spatial weights matrix may affect cluster detection.

Data:

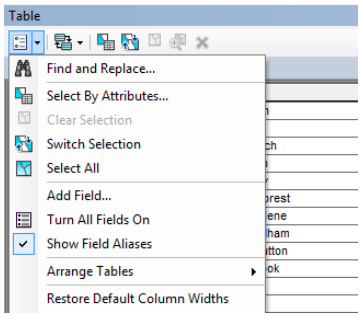
We will again use the male life expectancy data set for the East Midlands generated in a previous exercise (included here once more) and provided via this link: http://www.empho.org.uk/pages/viewResource.aspx?id=12351.

Activity instructions:

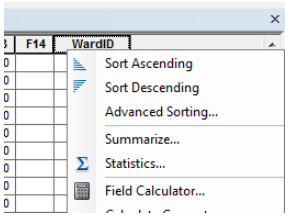
Pre-processing the map layer to prepare the spatial weights calculation

Open up the **malelifeexp** file from your earlier work in ArcMap, or use the one attached to this exercise. Having familiarised yourself with its content (e.g. by looking at its attribute table once more), right-click on its name in the left-hand table of contents and select properties, then take a look at the fields tab. To generate a spatial weights matrix, we need an integer attribute field that contains unique numbers for each feature in our file. However, this map layer currently does not have such a field, so we are first going to have to create one.

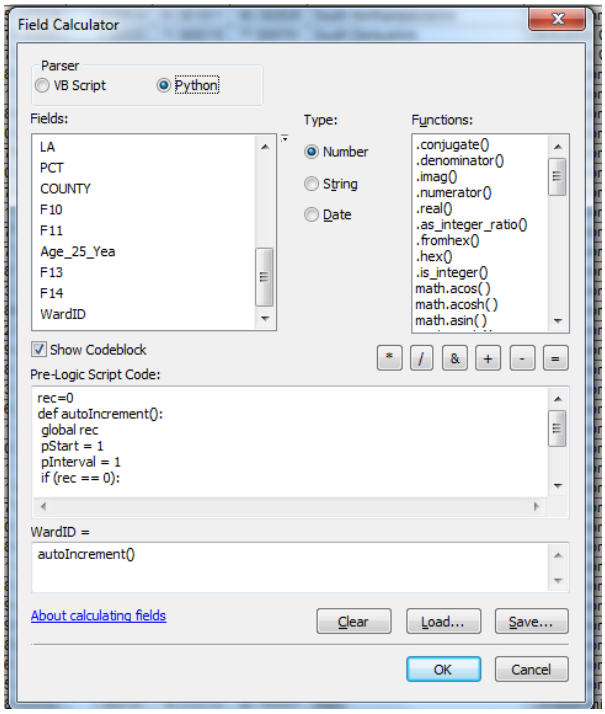
To do this, close down the properties window and then right-click on the layer again and select open attribute table. Choose *add field* from the menu in the top left corner and create a new field called **wardid** of type **longinteger**.



We now need to populate our new field with a unique ID number for each polygon. To do this, right-click on the header of our new **wardid** field and choose *field calculator:*



Unfortunately, there is no ready-made tool for calculating unique id numbers – at least not in a shape file – so we are going to have use a snippet of Python code (http://support.esri.com/fr/knowledgebase/techarticles/detail/38517). Set *parser* to be **python** and check the *codeblock* option:



This opens up a 'pre-logic script code' window, into which we can type some Python code for a function that will be run for each row in our table. A function is a self-contained piece of code that performs a particular task, in this case making a calculation and returning the result. If a task needs to be repeated, functions can be called many times. You should copy and paste the lines of code below into this new window:

```
rec=0
def autoIncrement():
 global rec
 pStart = 1
 pInterval = 1
 if (rec == 0):
  rec = pStart
 else:
  rec += pInterval
 return rec
```

What is this code actually doing? The annotation below explains what each line does:

```
rec=0 [a variable used to store the row number within the table, set to
zero before any records are processed]
def autoIncrement(): [this code defines a function called autoIncrement,
which is then called for each row processed in the table]
 global rec [this line gives our autoIncrement function permission to
change the value that is stored in the variable 'rec'. 'Rec' can be
accessed by the function itself, but also from code elsewhere too]
 pStart = 1  [two variables used only within our function – one being the
ID for the very first row,….]
```

```
 pInterval = 1 […. and the other indicating by how much the ID should
increase, each time a new row is processed]
 if (rec == 0):  [if it's the first row, set our row counter variable to be
our starting value]
  rec = pStart
 else:  [otherwise increase our row counter variable by one]
  rec += pInterval
 return rec [the value of the row counter variable gets passed back to the
place where the function is called.]
```

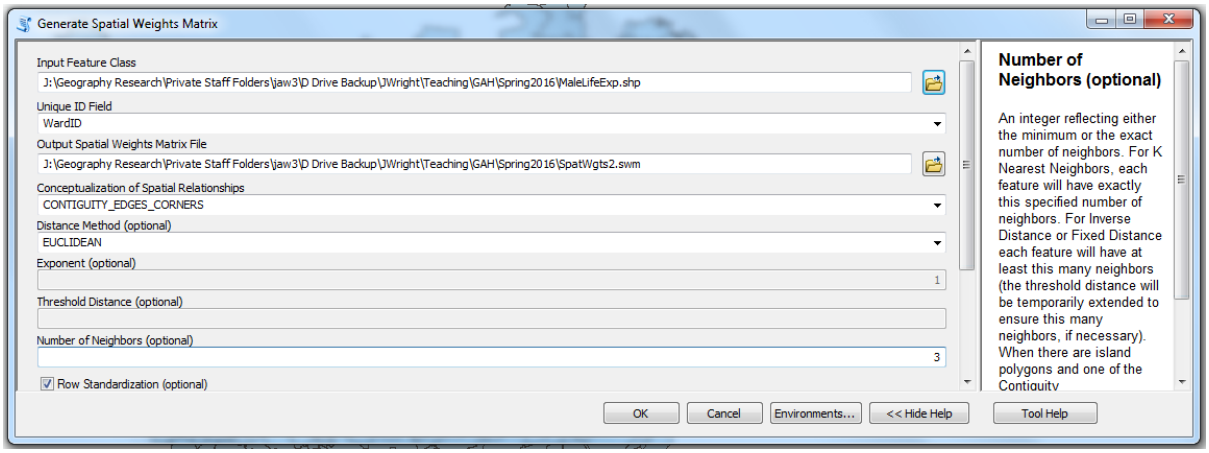
- Copy and paste this text into the *wardid* = box:

```
autoIncrement()
```

For every row in the table, our function will be run and then the result - `rec` – will be stored in our new field. Hit OK and you should (hopefully) end up with a set of unique values in **wardid**, one for each row.

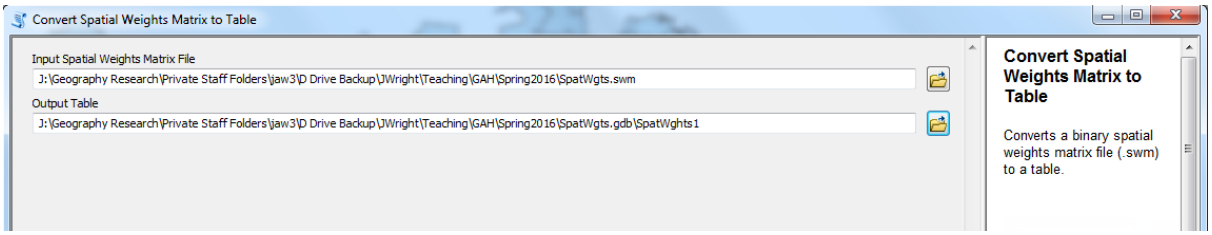
Calculating spatial weights

Now we have our unique IDs so we can keep track of each of our polygons, we will generate a spatial weights matrix. In the ArcToolBox, head for *spatial statistics tools*, then *modelling spatial relationships*, and then choose *generate spatial weights matrix*. As your *input feature class*, choose our **malelifeexp** file. We need a *unique ID field*, so we know which polygon is which, so choose **wardid** for this. Choose an output name for the spatial weights matrix – the weights are stored in a special binary format called a .swm file.



Now let us fill in the remaining options. For *conceptualization of spatial relationships*, select **contiguity_edges_corners**. What this will do is to generate spatial weights based on whether or not polygons are contiguous – in other words whether they share a common boundary or their boundaries touch at a given point. Notice the *number of neighbors* option. If we had an island in our study area, it would not be contiguous with <u>any</u> other polygons. This might be unrealistic – often islands share a lot of characteristics with neighbouring mainland areas via ferry routes for example. If we set this option to **1** (ignore my '3' in the screenshot!), should ArcGIS find a polygon that lacks any neighbours, it will automatically match it to the closest polygon. Let us leave the *distance method* as **Euclidean** (i.e. straight-line) for now and keep *row standardization* checked too (we will explain this in a moment). Hit OK and ArcGIS should generate the spatial weights matrix.

If we want to take a closer look at what ArcGIS has done and look behind the scenes, we need to convert the .swm file to a format where we can view it. To do this, head back to the *spatial statistics* area once more, select *utilities,* and then *convert spatial weights matrix to table.* Choose your newly created spatial weights matrix as the input file and for the *output table*, to avoid errors about file or pathway characters, we recommend storing this in a file geodatabase (so you may need to create a file geodatabase before specifying the table name).



ArcGIS should now create a new version of the spatial weights matrix, but in a standard table format rather than as a .swm file. Right-click on this new table and select *open* to view its contents – you should see something like this:

Table

SpatWghts1

OBJECTID *	WARDID	NID	WEIGHT
1	13	1	0.333333
2	13	9	0.333333
3	13	14	0.333333
4	9	10	0.333333
5	9	13	0.333333
6	9	14	0.333333
7	1	13	0.2
8	1	14	0.2
9	1	16	0.2
10	1	17	0.2
11	1	18	0.2
12	10	9	0.25

There are three fields (apart from a unique OBJECTID) attached to each row. One is **wardid**, our unique ID for each polygon. The second, **nid**, is basically the same unique ID again, but this time for a neighbouring polygon, so each row represents a polygon pair. The final field contains the **weight** for each polygon pair: the bigger the weight, the stronger the spatial connection between the pair of polygons. If a polygon pair is absent from the table, they do not share a common boundary and their weight will be considered zero. How are the weights calculated? If we take a look at the first polygon, with **wardid** = **13**, we can see what is happening. According to the table, there are three other polygons (with **wardid = 1, 9 and 14**) that share a boundary with this polygon with ID = 13.

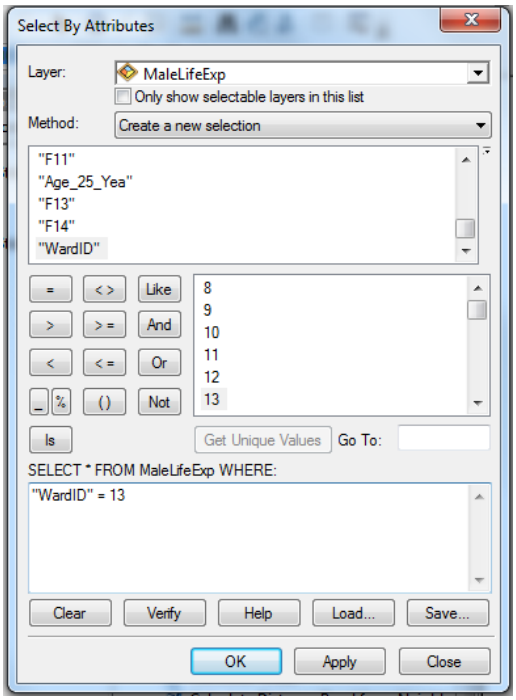
Rather than a table where each row represents a pair of polygons, we could actually structure the same information about the connections between this polygon and its neighbours differently. We could store this same information as a matrix (shown below with the information about our polygon with **wardid=13** highlighted in italics), where each polygon ID is a row heading and a column heading, and the cells of the table contain the spatial weights. Let us enter in a weight of 1 if a pair of polygons is contiguous, or zero otherwise (we do not enter any value if the row and column polygon ID are the same). If we added up all the weights looking across the rows of this table, we would find that the sum of weights for our polygon (with wardid=13) was 3. Recall that earlier, we ticked an option *row standardization*. If you tick this option, what happens is that in a final step, each cell value is divided by this row total. So, the weight between say polygon 13 and polygon 1 = 1 / 3 = 0.33. The effect of this is that where a polygon has only a single connection with another

polygon (e.g. a peninsula), that connection gets a high weight and is considered important. In contrast, the individual connections from a polygon that shares its boundaries with many neighbouring polygons receive much lower weights.

	NID				
WardID	**1**	**9**	**13**	**14**	**Row Total**
1	-	0	1	0	1
9	0	-	1	1	2
13	*1*	*1*	-	*1*	*3*
14	0	2	1	-	1
Column Total	1	2	3	1	7

Now take another look at the spatial weights matrix table in ArcGIS – see if you can square up this calculation with the way the weights are constructed in this table.

As a final step, we should check and make sure that ArcGIS has done the calculation correctly. Close down your spatial weights matrix table. Go to the *selection* menu and choose *select by attributes*. Use this tool to find the polygon that has **wardid = 13:**

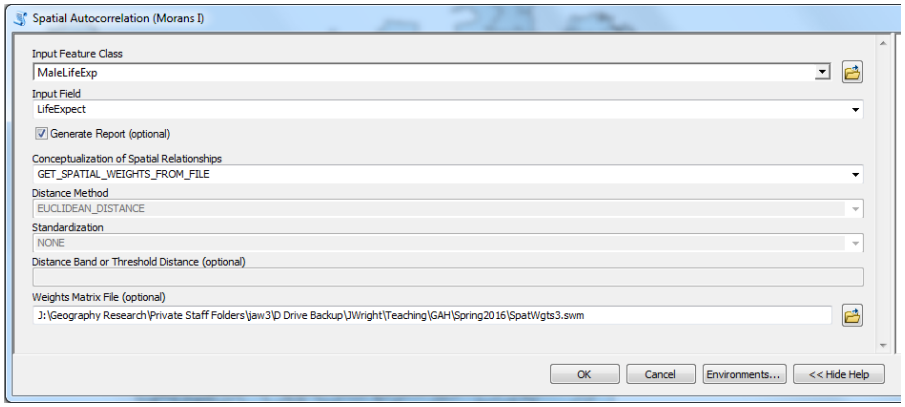


You should see this polygon highlighted. Now use the 'identify' tool to click around this polygon on the map – does the **wardid** of the neighbouring polygons match what you saw in the spatial weights matrix?

Now experiment with generating another spatial weights matrix using this same process, but using a different set of input parameters and see what happens to the spatial weights (Tip: be sure to right-click on your life expectancy layer and choose *selection / clear selection* before generating any more spatial weights matrices – otherwise, a spatial weights matrix for the single selected polygon with wardid = 13 would not be very helpful!)

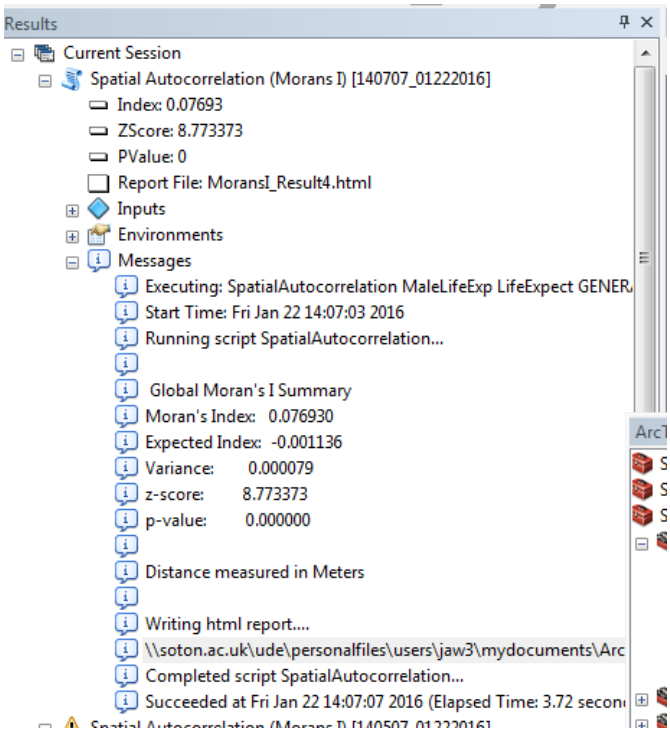
Impact of spatial weights matrices on global cluster detection

Now let us examine how the construction of the spatial weights matrix affects a test for clustering. Head for the spatial statistics area once more, but this time, let us choose *spatial statistics tools / analysing patterns / spatial autocorrelation (Moran's I)* from the ArcToolBox. This will calculate Moran's I for us. Let us pick up the our male life expectancy once more, and as the *input field*, let us choose **lifeexpect**, the field containing our male life expectancy numbers:



Under *conceptualisation of spatial relationships*, if you select **get_spatial_weights_from_file**, you should be able to pick up and use the .swm files that you created earlier. Run the tool and see what results you get.

Tip: If you go to the *geoprocessing* menu and select *results*, this will bring up the Results of your Moran's I calculation in a window. You need to click on the '+' next to messages to see the output. If you check 'generate report' when running the tool, notice also that an html format summary of your results will be generated (look under 'writing html report..' in the output to identify where this is located):



Experiment with different spatial weights – do they affect the global Moran's I test results?

If you look at the ArcGIS online help for (say) the global Moran's I test for spatial autocorrelation, you can find the formula for the test and work out what is happening behind the scenes:

The Moran's I statistic for spatial autocorrelation is given as:

$$I = \frac{n}{S_0} \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} w_{i,j} z_i z_j}{\sum_{i=1}^{n} z_i^2} \quad (1)$$

where z_i is the deviation of an attribute for feature i from its mean $(x_i - \bar{X})$, $w_{i,j}$ is the spatial weight between feature i and j, n is equal to the total number of features, and S_0 is the aggregate of all the spatial weights:

$$S_0 = \sum_{i=1}^{n} \sum_{j=1}^{n} w_{i,j} \quad (2)$$

We can actually see the numbers going into the calculation in our spatial weights table. Following ESRI's notation:

Table

SpatWghts1

OBJECTID *	WARDID	NID	WEIGHT
1	13	1	0.333333
2	13	9	0.333333
3	13	14	0.333333
4	9	10	0.333333
5	9	13	0.333333
6	9	14	0.333333
7	1	13	0.2
8	1	14	0.2
9	1	16	0.2
10	1	17	0.2
11	1	18	0.2
12	10	9	0.25

WARDID contains values of *i*….

NID contains values of *j*….

…and WEIGHT contains values of w_{ij}….

n is the number of polygons we have (maximum value in WARDID), whilst S_0 is the total of all the numbers in the WEIGHT field added together.

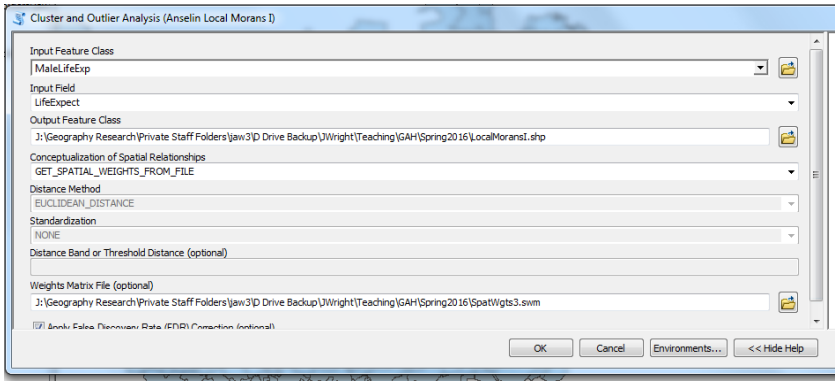
That covers all of the numbers in the calculation that reflect spatial relationships.

In this example, the remaining numbers (ESRI's z_i and z_j) in our calculation relate to life expectancy. They are the difference in life expectancy for a given polygon minus the average life expectancy across the whole study area. If you open up the life expectancy map layer attribute table, right-click on the **lifeexpect** field and select *statistics*, you can figure out the mean male life expectancy across the polygons in our study area – 78.45 years. If you used *field calculator* to subtract this mean of 78.45 from the values in the **lifeexpect** field, that would give you z_i. The result would be a positive number for a polygon with above-average life expectancy, but a negative number for a polygon with below-average life expectancy. For any pair of polygons, Moran's I involves multiplying the respective values together ($z_i z_j$). If life expectancy in both polygons is above average, multiplying them gives a positive result; if both have below average life expectancy, the result is again positive. However, the result is negative if one polygon has below average life expectancy and the other above average.

Because that number in turn gets multiplied by the spatial weight, the pairwise calculation $z_i z_j$ only counts towards the overall Moran's I value if the polygons are close together. That is why Moran's I becomes positive when neighbouring polygons have similar values.

Impact of spatial weights matrices on local cluster detection

If you head for *spatial statistics tools / analysing patterns / cluster and outlier analysis*, you will find an implementation of the local Moran's I test. Unlike the global Moran's I test, which gives an overall statistic that summarises the degree of clustering across an entire study area, this local test identifies the specific locations of clusters of elevated or reduced life expectancy in a data set. Try running this tool using the first of the spatial weights matrices that you created earlier:



Note: it is advisable to check the 'apply false discovery rate' correction, given that we are repeatedly applying a statistical test across many locations with this tool.

Before running it, have a quick look at the online help to see how this tool works. You will see that the local Moran's I uses a similar formula to global Moran's I, except that it effectively looks across just one row (*i*) of the spatial weights matrix:

The Local Moran's I statistic of spatial association is given as:

$$I_i = \frac{x_i - \bar{X}}{S_i^2} \sum_{j=1, j \neq i}^{n} w_{i,j}(x_j - \bar{X}) \qquad (1)$$

where x_i is an attribute for feature i, $\bar{X}$ is the mean of the corresponding attribute, $w_{i,j}$ is the spatial weight between feature i and j, and:

$$S_i^2 = \frac{\sum_{j=1, j \neq i}^{n} (x_j - \bar{X})^2}{n - 1} - \bar{X}^2 \qquad (2)$$

with n equating to the total number of features.

If we ignore the term S^2_i for a moment, you can see that $x_i - X$ is once again the life expectancy for a given ward, minus the mean life expectancy across the entire study area. $W_{i,j}$ are our weights, which will be zero for polygons that do not share a boundary with our given ward i. For a given polygon, we take each neighbouring ward j's life expectancy, then subtract the mean life expectancy across the entire study area from this figure $(x_j - X)$. We then add the resultant numbers together for all neighbouring polygons. Non-neighbouring polygons, which have weights of zero, are ignored in this calculation. Finally, we multiply this by the difference between life expectancy in our chosen ward i and life expectancy across the entire study area. Other than the additional step involving S^2_i, that number is the local Moran's I value. Taking these two numbers, we can pick out four different situations:

- If life expectancy is above average in our chosen ward i and is above average for its neighbours j, the local Moran's I value will be positive – we will be multiplying a positive number by another positive number. That is a 'high-high' cluster, a spatial concentration of high life expectancy values.
- If life expectancy is below average in our chosen ward i and is below average for its neighbours j, the local Moran's I value will be positive – because we will be multiplying a negative number by another negative number. That is a 'low-low' cluster, a spatial concentration of low life expectancy values.
- If life expectancy is above average in our chosen ward i but below average for its neighbours j, the local Moran's I value will be negative – we will be multiplying a positive number by a negative number. That is a 'high-low' outlier, a ward with high life expectancy surrounded by low life expectancy wards.
- If life expectancy is below average in our chosen ward i but above average for its neighbours j, the local Moran's I value will be negative – we will be multiplying a negative number by a positive number. That is a 'low-high' outlier, a ward with low life expectancy surrounded by high life expectancy wards.

Now hit OK and take a look at the output. This time, ArcGIS will create a new output map layer that contains information about clusters and (if they exist as often these are rarer) outliers around each location.

If you take your output map layer and right-click on it in the left-hand panel, then choose *open attribute table*, you can see the fields created for each polygon, alongside the original life expectancy data and **wardID** numbers. **LMiindex** contains the local Moran's I index values; **COType** indicates (where significant) the nature of any cluster or outlier, following the typology described above. **LMiZScore** is a measure of the extent to which the calculated local Moran's I value differs from its expected value under spatial randomness, known as the Z score. Zero indicates the value is exactly as expected; a large negative value suggests an outlier; whilst a large positive value suggests a cluster. **LMiPValue** indicates the probability (between 0 and 1) that the given local Moran's I value could have arisen by chance. P values very close to zero are significant and highly unlikely to have arisen by chance, whilst the closer the Z score lies to 0, the more likely it is to have arisen by chance and the P-values approach 1. Do the patterns and data look plausible?

Finally, as you did before, try out one other set of spatial weights and see how this affects your output. Does the pattern of clusters detected change at all?

More on spatial weights:

If there are different ways of generating spatial weights, which method should we choose? Although much depends on the specific problem in hand, you can find additional tips on spatial weights here – see the section on 'best practice' towards the bottom of this page: http://webhelp.esri.com/arcgisdesktop/9.3/index.cfm?TopicName=Modeling_spatial_relationships

Note that you can create spatial weights in even more sophisticated ways than we have covered here. Our weights were based on *Euclidean* (straight-line distances). However, it is possible to use non-Euclidean distances such as drive-times when creating spatial weights. Although beyond the scope of this module, ArcGIS includes a tool called *generate network spatial weights,* which can be used to generate weights based on travel times rather than Euclidean distances. Similarly, *generate*

spatial weights matrix has a *convert table* option, which can be used to produce an ESRI format .swm file from a table produced using (say) third party drive-time software (effectively doing the reverse of *convert spatial weights matrix to table* and turning a table into a .swm file rather than vice versa).

One final point: the spatial weights matrix is at the heart of a great many spatial analysis operations, including specialised, spatial forms of regression analysis as well as cluster detection. Your knowledge of spatial weights matrices should therefore be useful for other forms of spatial analysis too.