9 Basic Spatial Analyses

Introduction

Spatial data analysis is the application of operations to coordinate and related attribute data, often to solve a problem. We may wish to identify high crime areas, to generate a list of road segments that need repaving, or find the best area to place wind turbines. There are hundreds of *spatial operations* or *spatial functions* used in spatial analysis, and all involve calculations with coordinates or attributes.

Spatial operations are often applied sequentially to solve a problem, the output of each spatial operation serving as the input of the next (Figure 9-1). Part of the challenge in geographic analysis is selecting appropriate spatial operations, and applying them in the appropriate order.

The table manipulations we described in Chapter 8 are included in our definition of a spatial operation. Indeed, the selection and modification of attribute data in spatial data layers are included at some time in nearly all complex spatial analyses. Many operations incorporate both the attribute and coordinate data, and the attributes must be further selected and modified in the course of a spatial analysis.

The discussion in the present chapter will expand on rather than repeat the selection operations treated in Chapter 8. This chapter describes spatial data analyses that involve sort, selection, classification, and spatial operations that are applied to both coordinate and associated attribute data.

Input, Operations, and Output

Spatial data analysis typically involves using data from one or more layers to create output. The analysis may consist of a single operation applied to a data layer, or many operations that integrate input data from many layers to create the desired output.

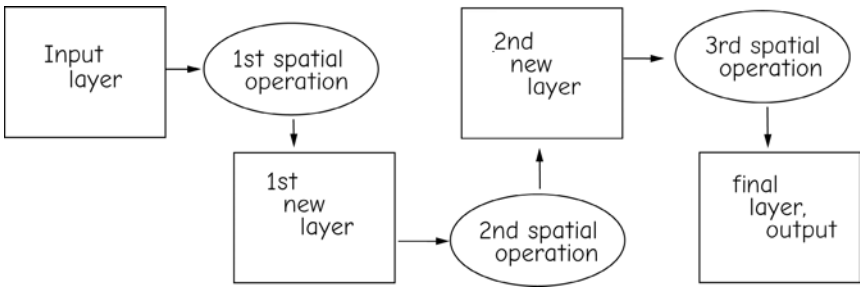


Figure 9-1: A sequence of spatial operations is often applied to obtain a desired final data layer.

There are also operations that generate several output data layers from a single input. Terrain analysis functions may take a raster grid of elevations as an input data layer and produce both slope (local steepness) and aspect (the slope direction). In this case, two outputs are generated for each input elevation data set.

Operations may also take several input layers to generate a single output layer. A layer average is an example of the use of multiple input layers to produce a single output layer, for example, annual rainfall found by summing 12 monthly rainfall raster layers. Finally, there are some spatial operations that require many input layers and generate many output layers.

The output from a spatial operation may be spatial, creating new spatial data layers, or nonspatial, producing scalar values or a table, with no explicit geometric data attached. A layer average function may simply calculate the mean cell value found in a raster data layer. The input is a spatial data layer, but the output is a single number.

Scope

Spatial data operations may be characterized by their *spatial scope*, the extent or area of the input data that are used in determining the values at output locations (Figure 9-2). Spatial operations may be characterized as local, neighborhood, or global, to reflect the extent of the input area used to determine the value at a given output location.

Local operations use only the data at one input location to determine the value at that same output location (Figure 9-2, top). Attributes or values at adjacent locations are not used in the operation.

Neighborhood operations use data from both an input location plus nearby locations to determine the output value (Figure 9-2, center). The extent and relative importance of values in the nearby region may vary, but the value at an output location is influenced by more than just the value of data found at the corresponding input location.

Global operations use data values from the entire input layer to determine each output value. The value at each location depends in part on the values at all input locations (Figure 9-2, bottom).

The set of available spatial operations depends on the data model and type of spatial data used as input. Some operations may be easily applied to raster or vector data. While the details of the specific implementation may change, the concept of the operation does not. Other operations may be possible in only one data model.

Characteristics of a data model will determine how any given operation is applied. The specific implementation of many operations, for example, multilayer addition, depends on the specific data model. A raster operation may produce a different outcome than a vector operation, even if the themes are meant to represent the same features. In a like manner, the specific set and sequence of operations in a spatial analysis will depend on the data model used and the specific operations available in the GIS software.

Spatial scope provides a good example of this influence of data models. Cells in a raster data set have uniform size and shape. A local operation applied to a raster data layer has a well-defined, repeatable area. In contrast, polygons usually vary in size and shape. A local operation for a vector polygon data set is likely to have variable size and shape from one location to another. In Figure 9-2, the local operation follows a state boundary. Therefore, the operation applies to a different size and shape for each state.

Neighborhood analyses are affected by the shape of adjacent states in a similar manner. Summary values such as populations of adjacent states may be greatly influenced by changes in neighborhood size, so great care must be taken when

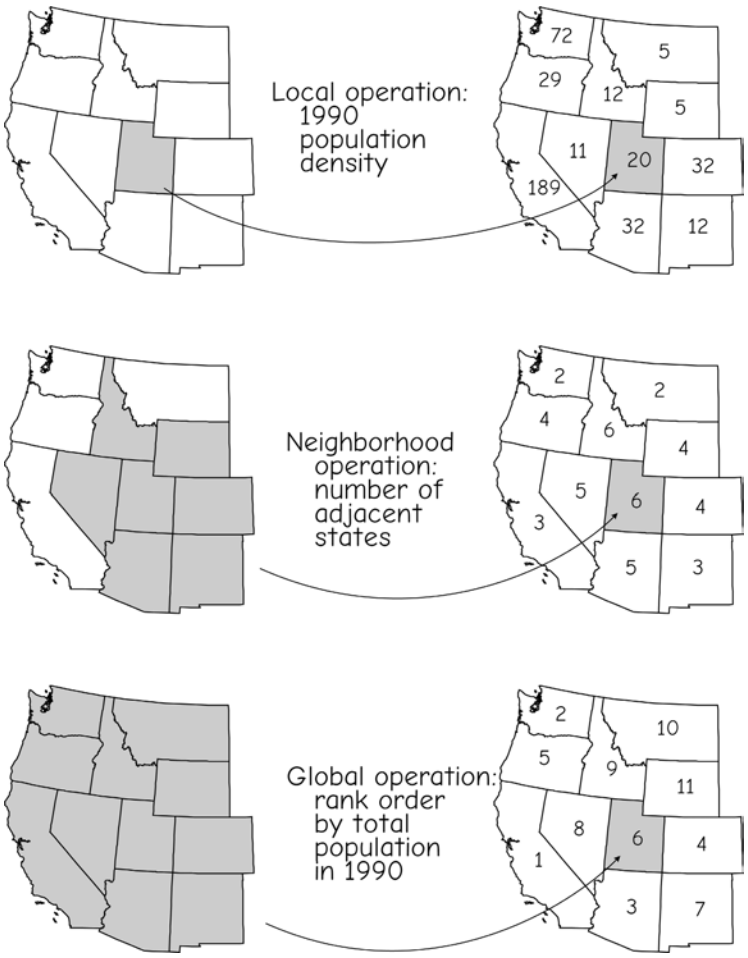


Figure 9-2: Local, neighborhood, and global operations. Specific input and output regions are shown for Utah, the shaded area on the right side of the figure. Shaded areas on the left contribute to the values shown in the shaded area on the right. Local operation output (top right) depends only on data at the corresponding input location (top left). Neighborhood operation output (middle right) depends on input from the local and surrounding areas (middle left). Global operation output (bottom right) depends on all features in the input data layer (bottom left).

interpreting the results of a spatial operation. Knowledge of the algorithm behind the operation is the best aid to interpreting the results.

While most operations might be conceptually compatible with most spatial data models, some operations are easier to apply in some models. Most neighborhood operations are quite easy to program when using raster data models, and quite difficult when using vector data models. The reverse is true for network operations, which are generally easier to apply in vector models. In many instances, it is more efficient to convert the data between data models and apply the desired operations and, if necessary, convert the results back to the original data model.

Selection and Classification

Selection operations identify features that meet one to several conditions or criteria. In these operations, attributes or geometry of features are checked against criteria, and those that satisfy the criteria are selected. These selected features may then be written to a new output data layer, or the geometry or attribute data may be manipulated in some manner.

Figure 9-3 shows an example of a selection operation that involves the attributes of a spatial data set. Two conditions are applied, and the features that satisfy both conditions are included in the selected set.This example shows the selection of those states in the "lower 48" United States that are a) entirely north of Arkansas, and b) have an area greater than 84,000 km^2. The complete set of features that will be considered is shown at the top of the figure. This set is composed of the lower 48 states, with the state of Arkansas indicated by shading. The next two maps of Figure 9-3 show those states that match the individual criteria. The second map from the top shows those states that are entirely north of Arkansas, while the third map shows all those states that are greater than 84,000 km^2. The bottom part of Figure 9-3 shows those states that satisfy both conditions. This figure illustrates two basic characteristics of selection operations. First, there is a set of features that are candidates for selection, and second, these features are selected based singly or on some combination of the geographic and attribute data.

The simplest form of selection is an *on-screen query*. A data layer is displayed, and features are selected by a human operator. The operator uses a pointing device to locate a cursor over a feature of interest and sends a command to select, often via a mouse click or keyboard entry. On-screen (or interactive) query is used to gather information about specific features, and is often used for interactive updates of attribute or spatial data. For example, it is common to set up a process such that when a feature is selected, the attribute information for the feature is displayed. These attribute data may then be edited and the changes saved.

Queries may also be specified by applying conditions solely to the aspatial components of spatial data. These selections are most often based on the attribute data tables for a layer or layers. These selection operations are applied to a set of features in a data layer or layers. The attributes for each feature are compared to a set of conditions. If the attributes match the conditions, they are selected; if the attributes fail to match the conditions, they are placed in an unselected set. In this manner, selection splits the data into either the selected set or the unselected set. The selected data are then typically acted on in some way, often saved to a separate file, deleted, or changed in some manner.

Selection operations on tables were described in general in Chapter 8. The description here expands on that information and draws attention to specific characteristics of selections applied to spatially related data. Table selections have spatial relevance because each record in a table is associated with a geographic feature. Selecting a record in a table simultaneously selects the associated spatial features: cells, points, lines, or areas. Spatial selections may be combined with table selections to identify a set of selected geographic features.

Set Algebra

Selection conditions are often formalized using *set algebra*. Set algebra uses the operations less than (<), greater than (>), equal to (=), and not equal to (< >). These selection conditions may be applied either alone or in combination to select features from a set.

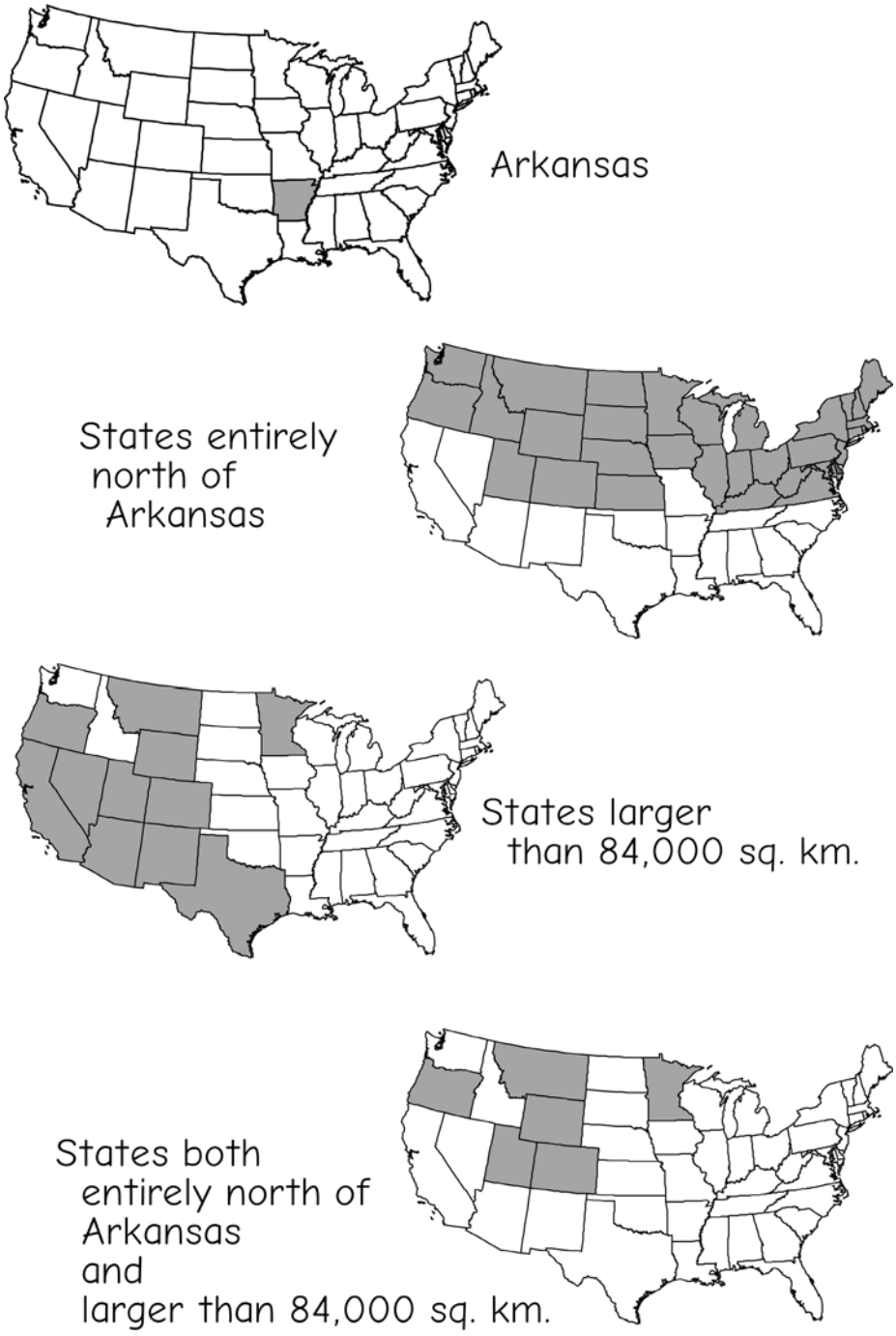


Figure 9-3: An example of a selection operation based on single or multiple conditions.

Figure 9-4 shows four set algebraic expressions and the selection results for a set of counties in the northeastern United States. The upper two selections show equal to (=) and not equal to (< >) selections. The upper left shows all counties with a value for the attribute state that equals Vermont, while the upper right shows all counties with a value for state that are not equal to New York. The lower selections in Figure 9-4 show examples of ordinal comparisons. The left figure shows all counties with a size greater than or equal to (> =) 1,000 mi^2, while the right side shows all counties with a population density less than (<) 250 persons per mi^2.

The set algebra operations greater than (>) or less than (<) may not be applied to nominal data, because there is no implied order in nominal data. Green is not greater than yellow, and red is not less than blue. Only the set algebra operations equal to (=) and not equal to (< >) apply to these nominal variables. All set algebra operations may be applied to ordinal data, and all are often applied to interval/ratio data.

Boolean Algebra

Boolean algebra uses the conditions OR, AND, and NOT to select features. Boolean expressions are most often used to combine set algebra conditions and create compound selections. The Boolean expression consists of a set of Boolean operators, variables, and perhaps constants or scalar values.

Boolean expressions are evaluated by assigning an outcome, true or false, to each condition. Figure 9-5 shows three examples of Boolean expressions. The first is an expression using a Boolean AND, with two arguments for the expression. The first argument specifies a condition on a variable named area, and the second argument a condition on a variable named farm_income. Features are selected if they satisfy both arguments, that is, if their area is larger than 100,000 AND farm_income is less than 10 billion.

Expression 2 in Figure 9-5 illustrates a Boolean NOT expression. This condition specifies that all features with a variable state which is not equal to Texas will return a true value, and hence be selected. NOT is also often known as the negation operator. This is because we might interpret the application of a NOT operation as exchang-

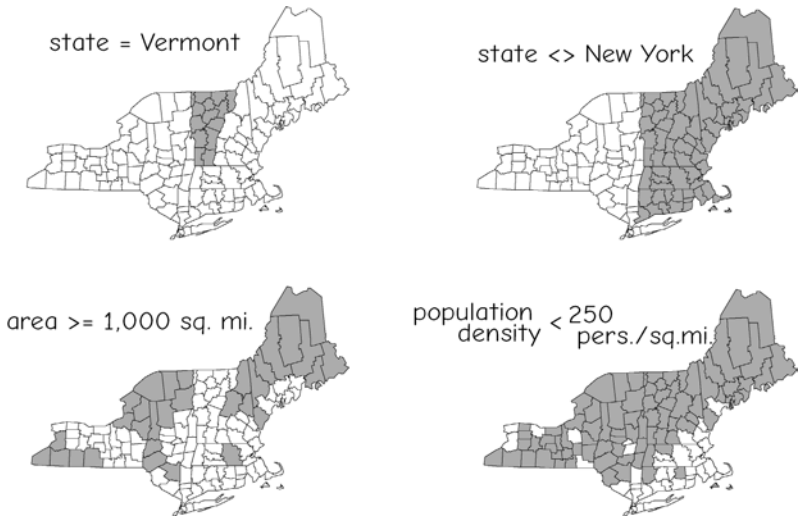


Figure 9-4: Examples of expressions in set algebra and their outcome. Selected features are shaded.

Boolean expressions

1. (area > 100,000)
   AND
   (farm_income < 10 billion)

2. NOT (state = Texas)

3. [(rainfall > 1,000)
     AND
     (taxes = low)]
   OR
   [(house_cost < 65,000)
     AND
     NOT (crime = high)]

Figure 9-5: Examples of Boolean expressions.

ing the selected set for the unselected set. The argument of expression 2 in Figure 9-5 is itself a set algebra expression. When applied to a set of features, this expression will select all features for which the variable state is equal to the value Texas. The NOT operation reverses this, and selects all features for which the variable state is not equal to Texas.

The third expression in Figure 9-5 shows a compound Boolean expression, combining four set algebra expressions with AND, OR, and NOT. This example shows what might be a naive attempt to select areas for retirement. Our grandparent is interested in selecting areas that have high rainfall and low taxes (a gardener on a fixed income), or low housing cost and low crime.

The spatial outcomes of specific Boolean expressions are shown in Figure 9-6. The figure shows three overlapping circular regions, labeled A, B, and C. Areas may fall in more than one region; for example, the center, where all three regions overlap, is in A, B, and C. As shown in the figure, Boolean AND, OR, or NOT may be used to select any combination or portions of these regions.

OR conditions return a value of true if either argument is true. Areas in either region A or region B are selected at the top center of Figure 9-6. AND requires the conditions on both sides of the operation be met; an AND operation results in a reduced selection set (top right, Figure 9-6). NOT is the negation operator, it flips the effect of the previous operations; it turns true to false and false to true. The NOT shown in the lower left portion of Figure 9-6 returns the area that is only in region C. Note that this is the converse, or opposite set that is returned when using the comparative OR, shown in the top center of Figure 9-6. The NOT operation is often applied in combination with the AND Boolean operator, as shown at the bottom center of Figure 9-6. Again, this selects the converse (or com-

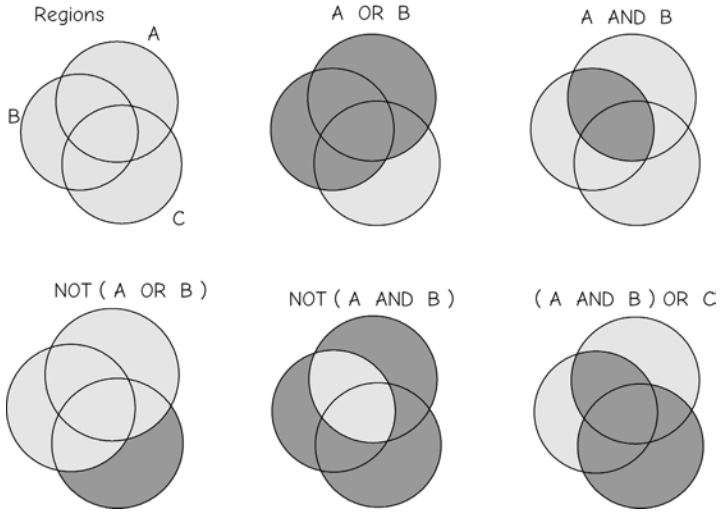


Figure 9-6: Examples of expressions in Boolean algebra, and their outcomes. Subareas of three regions are selected by combining AND, OR, and NOT conditions in Boolean expressions. Any sub-area or group of sub-areas may be selected by the correct Boolean combination.

(County = Rice)
AND
(Wshed = Cannon)

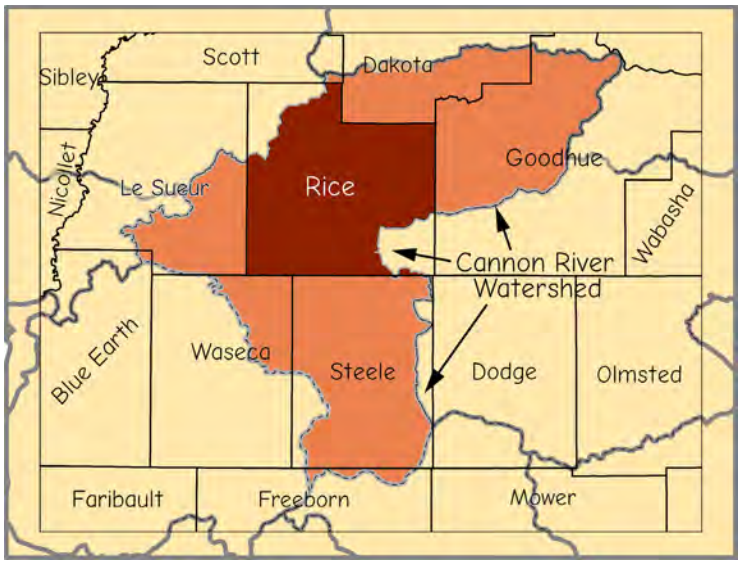


Figure 9-7: An example of a Boolean selection applied to a set of counties and watersheds in the midwestern United States. The mid shaded area is within the Cannon River watershed, a tributary of the Mississippi River, while the darkest shaded area is also within Rice County, Minnesota.

plement) of the corresponding AND. Compare the bottom center selection to the top right selection in Figure 9-6. NOTs, ANDs, and ORs may be further combined to select specific combinations of areas, as shown in the lower right of Figure 9-6.

Note that as with table selection discussed in Chapter 8, the order of application of these Boolean operations is important. In most cases, you will not select the same set when applying the operations in a different order. Parentheses, brackets, or other delimiters should be used to specify the order of application. The expression A AND B OR C will give different results when interpreted as (A AND B) OR C, as shown in Figure 9-6, than when interpreted as A AND (B OR C). Verify this as an exercise. Which areas does the second Boolean expression select?

Figure 9-7 shows a real-world example of a Boolean selection. Counties often must identify areas for treatment, in this case a portion of the Cannon River, a tributary of the Mississippi River, targeted for pollution reduction. Counties are labeled, with boundaries shown as thick solid lines. The Cannon River watershed is shown in darker shades of gray. A Boolean AND operation was applied to a data layer containing both watershed and county boundaries, selecting the areas that are both within the Cannon River watershed and within Rice County.

Spatial Selection Operations

Many spatial operations select sets of features. These operations are applied to a spatial data layer and return a set of features that meet a specified condition. Adjacency and containment are commonly used spatial selection operations.

Adjacency selection operations are used to identify those features that "touch" other features. Features are typically considered to touch when they share a boundary, as when two polygons share an edge. A target or key set of polygon features is identified, and all features that share a boundary with the target features are placed in the selected set.

Figure 9-8a shows an example of a selection based on polygon adjacency. The state of Missouri is shaded on the left side of Figure 9-8a, and states adjacent to Missouri are shaded on the right portion of Figure 9-8a. States are selected because they include a common border with Missouri.

There are many ways the shared border may be detected. With a raster data layer, an exhaustive cell-by-cell comparison may be conducted to identify adjacent pairs with different state values. Vector adjacency may be identified by observing the topological relationships (see Chapter 2 for a discussion of topology). Line and polygon topology typically records the polygon identifiers on each side of a line. All lines with Missouri on one side and a different state on the other side may be flagged, and the list of states adjacent to Missouri extracted.

Adjacency is defined in Figure 9-8a as sharing a boundary for some distance greater than zero. Figure 9-8b shows how a different definition of adjacency may affect selection. The left of Figure 9-8b shows the state of Arizona and a set of adjacent (shaded) western states. By the definition of adjacency used in Figure 9-8a, Arizona and Colorado are not adjacent, because they do not share a boundary along a line segment. Arizona and Colorado share a border at a point, called Four Corners, where they join with Utah and New Mexico. When a different definition of adjacency is used, with a shared node qualifying as adjacent, then Colorado is added to the selected adjacent set (right, Figure 9-8b). This is another illustration of an observation made earlier; there are often several variations of any single spatial

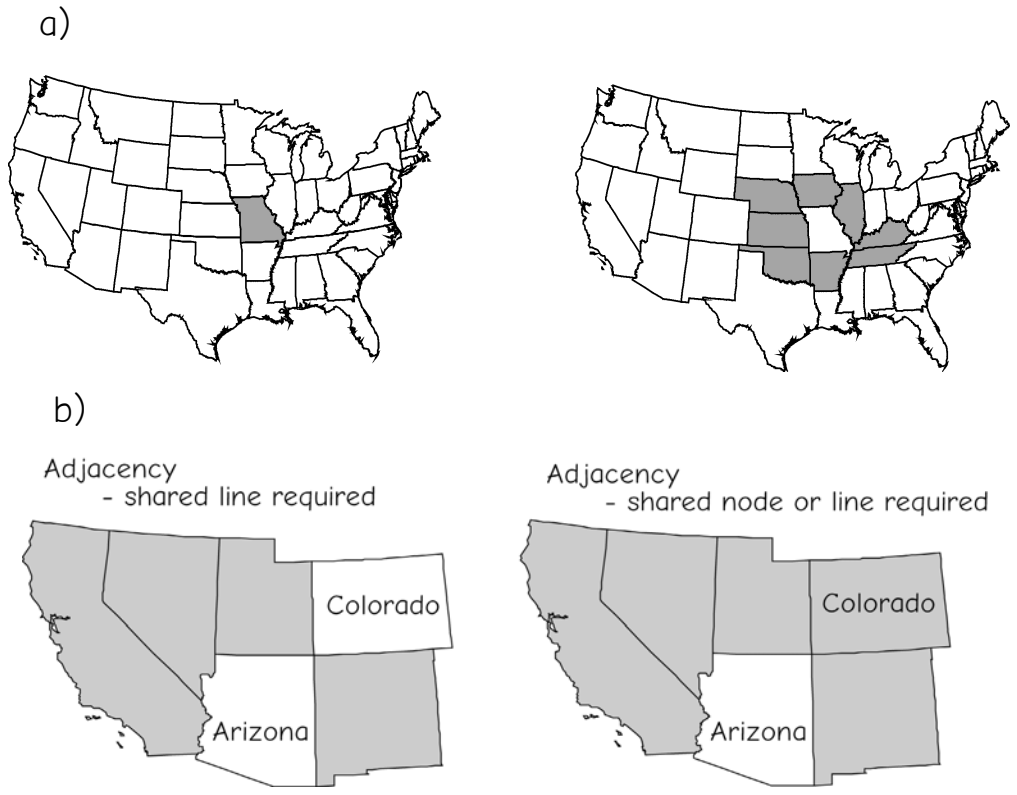


Figure 9-8: Examples of selections based on adjacency. a) Missouri, USA is shown on the left and all states adjacent to Missouri shown on the right. b) Different definitions of adjacency result in different selections. Colorado is not adjacent to Arizona when line adjacency is required (left), but is when node adjacency is accepted (right).

operation. Care must be taken to test the operation under controlled conditions until the specific implementation of a spatial operation is well understood.

Features may be selected based on proximity. Proximity selection typically requires a set of selecting features and a set of target features. All the target features within a specified distance of the selecting features will be chosen, e.g., all weather stations within 60 km of a watershed may be selected to estimate rainfall (Figure 9-9). Selecting only stations within the watershed may provide poor estimates of rainfall near the edge of the polygon, but using all gages is inefficient because information from gages very far away often isn't helpful. An adequate exterior proximity is chosen, usually by visual inspection, previous experience, or preliminary tests, and the proximity selection applied. Proximity is usually calculated implicitly within the operation, in that first the source features and selectable features are provided, a proximity distance and method specified, and on running the operation, the selected set is identified. Less frequently, softwares require a multi-step process in which the source features and selection layers area identified, and a selection layer is created. This created layer usually contains polygons defining the area within the specified distance of the source features. This polygon layer is then used to select features from the target set.

There are several variants of proximity selection. One variant selects all features that are at least partially within a given distance of a set of features. Another variant selects only features that are entirely inside a given distance of the outer boundaries of a set of polygons. A third variant selects only those features that are entirely within a given distance of a set of polygons, but not those that are within the set of polygons themselves. Users should clearly identify the selection tool function and outputs.

Proximity selection, and most selection processes, typically only select features that meet a given set of criteria. The process often does not create a separate, new data layer of the selected features. Selected features are marked on-screen and in the corresponding data table, but still are part of the source data set. There is often an additional export step if these selected features are destined for a new data set. This explicit creation approach is usually

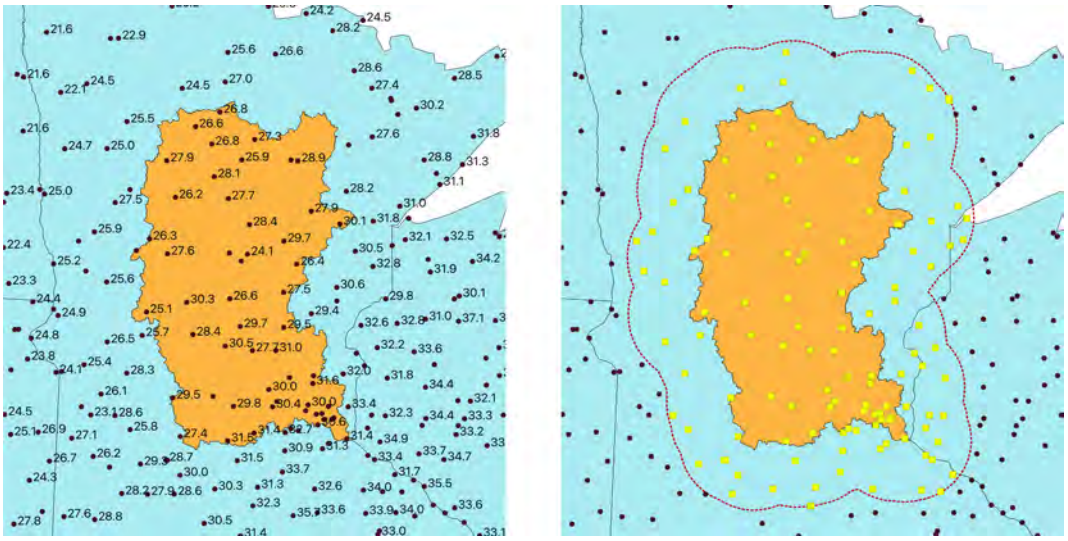


Figure 9-9: An example of a spatial selection based on proximity, where rainfall measured at gages (points, left panel, with mean label) near a watershed are selected. All gages are identified within a 60 km proximity of the watershed boundary (right panel, light squares).

adopted because selection is often a multi-step process, with various different selection tools applied successively to arrive at a target set of features.

We often add indicator or classification variables to our data tables when we have a complex set of selection criteria, particularly when combining both spatial and tabular selections. These indicator variables record the membership of features in groups that match or don't match a set of conditions. Figure 9-10 illustrates a selection for a set of counties that both contain part of a target watershed and hazardous materials (Hazmat) storage sites. A municipality might undertake an upstream inventory if they measure a spike in toxic chemicals in their water supply. Hazmat sites are distributed throughout the state of Georgia. The Altamaha River drains much of the central portion of the state. A spatial selection identifies counties that contain a portion of the watershed, and a table selection, described in Chapter 8, applied to the selected counties reduces the set to those that might be contributing to downstream pollution from Hazmat sites. A column may be added to the county table and values assigned to record this set of potential source counties, to be used in subsequent analysis.

Figure 9-11 shows one flowchart of the geoprocessing analysis in Figure 9-10. While the analysis is relatively simple, and the steps primarily operations on the tabular data, the flowchart shows the data, spatial operations, and order in a succinct manner. It is good practice to flowchart all multi-step spatial analyses, and the value of the flowchart grows with the complexity of the analysis.

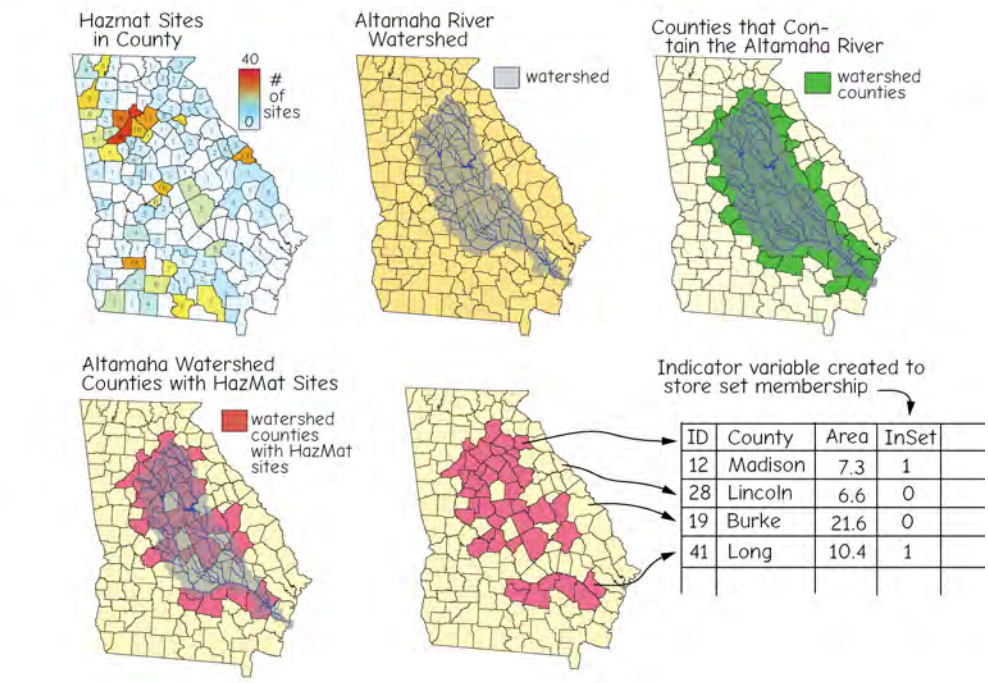


Figure 9-10: An example of indicator variables to record set membership. We wish to identify counties with hazardous materials sites (Hazmat, upper left) that contain part of the Altamaha River watershed (upper center). We may first apply a spatial selection on counties with the watershed boundary (upper right), then a table selection on the coincident counties to identify those with Hazmat sites (lower left). A column, here named InSet, may be added to identify the selected counties in further processing (bottom center and right).

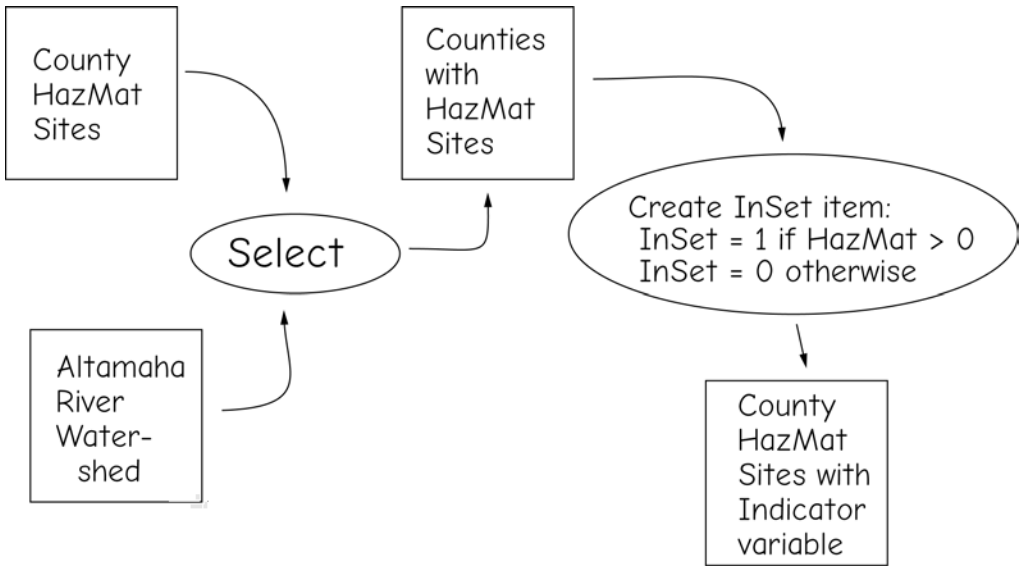


Figure 9-11: An example flowchart for the Altamaha HazMat site analysis. While simple, this shows how spatial and tabular operations might be represented graphically in a diagram.

Caution is helpful when applying subsequent spatial operations to a data set with selected features, e.g., the rainfall gages in the right panel of Figure 9-9. I might recalculate values in the attribute table, copy features, or apply another spatial operation. Some operations by default only act on selected features, while other operations apply to the entire data set. The choices are software-dependent, and so you should consult the documentation or test each new spatial operation when first using it to avoid unintended results

Containment is another spatial selection operation. Containment selection identifies all features that contain or surround a set of target features. For example, the California Department of Transportation may wish to identify all counties, cities, or other governmental bodies that contain some portion of Highway 99, because they wish to improve road safety. A spatial selection may be used to identify these governmental bodies.

Figure 9-12 illustrates a containment selection based on the Mississippi River in North America. We wish to identify states that contain some portion of the river and its tributaries. A query is placed, identifying the features that are contained, here the Mississippi River network, and the target features that may potentially be selected. The target set in this example consists of the lower 48 states of the United States. All states that contain a portion of the Mississippi River or its tributaries are shaded as part of the selected set.

Classification

Classification is a spatial data operation that is often used in conjunction with selection. A classification, also known as a *reclassification* or *recoding*, will categorize geographic objects based on a set of conditions. For example, all the polygons larger than one square mile may be assigned a `size` value equal to `Large`, all polygons from 0.1 to 1 square mile may be assigned a `size` equal to `Mid`, and all polygons smaller than 0.1 square miles may be assigned a `size` equal to `Small` (Figure 9-13). Classifications may add to or modify the attribute data for each geographic object. These modified attributes may in turn be used in further analyses, such as for more complex combinations in additional classification.

Figure 9-12: An example of a selection based on containment. All states containing a portion of the Mississippi River or its tributaries are selected.

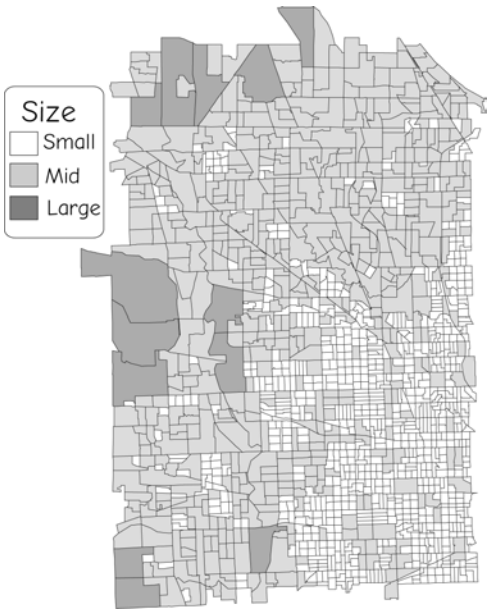


Figure 9-13: Land parcels re-classified by area.

Classification may be used for many other purposes. One common end is to group objects for display or map production. These objects have a common property, and the goal is to display them with a uniform color or symbol so the similar objects are identified as a group. The display color and/or pattern is typically assigned based upon the values of an attribute or attributes. A range of display shades may be chosen, and corresponding values for a specific attribute assigned. The map is then displayed based on this classification.

A classification may be viewed as an assignment of features from an existing set of classes to a new set of classes. We identify features that have a given set of values, for example, parcels that are above a certain size, and assign them all a classification value, in this case the class “large.” Parcels in another range of sizes may be assigned different class values, for example, “mid” and “small.” The attribute that stores the parcel area is used as a guide to assigning the new class value for size.

The assignment from input attribute values (area) to new class values (here, size) may be defined manually, or the assignment may be defined automatically. For manual classifications, the class transitions are specified entirely by the human analyst.

Classifications are often specified by a table or array. The table identifies the input class or values, as well as the output class for each of this set of input values. Figure 9-14 illustrates the use of a classification table to specify class assignment. Input values of `A` or `B` lead to an output class value of `1`, an input value of `E` leads to an output value of `2`, and an input value of `I` leads to an output value of `3`. The table provides a complete specification for each classification assignment.

Figure 9-14 illustrates a classification based on a manually defined table. A human analyst specifies the `In` items for the source data layer via a classification table, as well as the corresponding output value for each `In` variable. `Out` values must be specified for each input value or there will be undefined features in the output layer. Manually defining the classification table provides the greatest control over class assignment. Alternatively, classification tables may be automatically assigned, in that a number of classes may be specified and some rule embodied in a computer algorithm used to assign output classes for each of the input classes.

A *binary classification* is perhaps the simplest form of classification. A binary classification places objects into two classes: 0 and 1, true and false, `A` and `B`, or some other two-level classification. A set of features is selected and assigned a value, and the complement of the set, all remaining features in the data layer, is assigned the different binary value.

A binary classification is often used to store the results of a complex selection operation. A sequence of Boolean and set algebra expressions may be used to select a set of features. A specific target attribute is identified for the selected set of features.

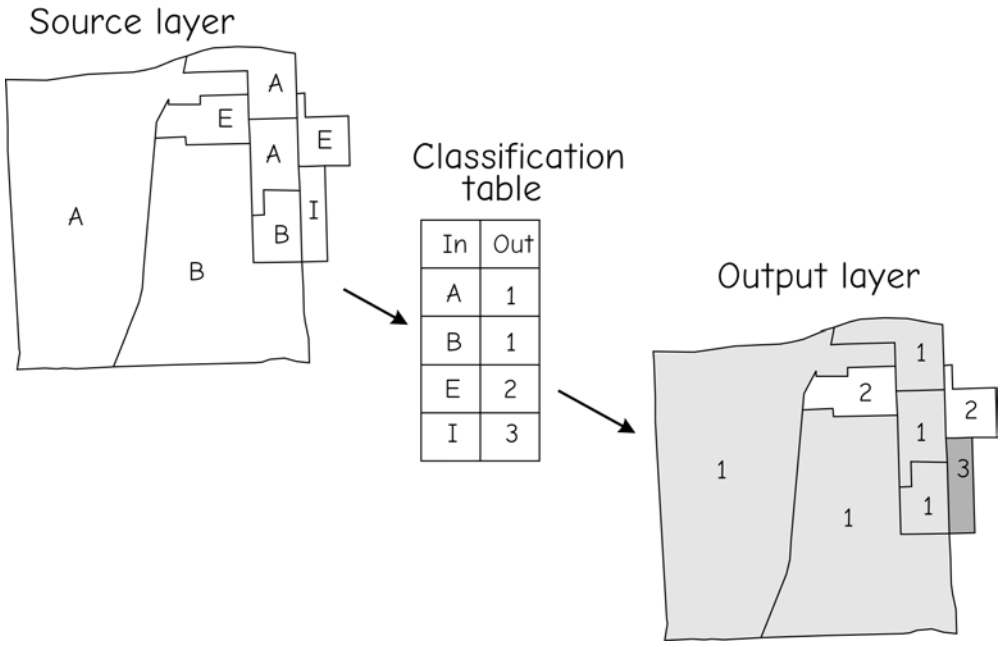
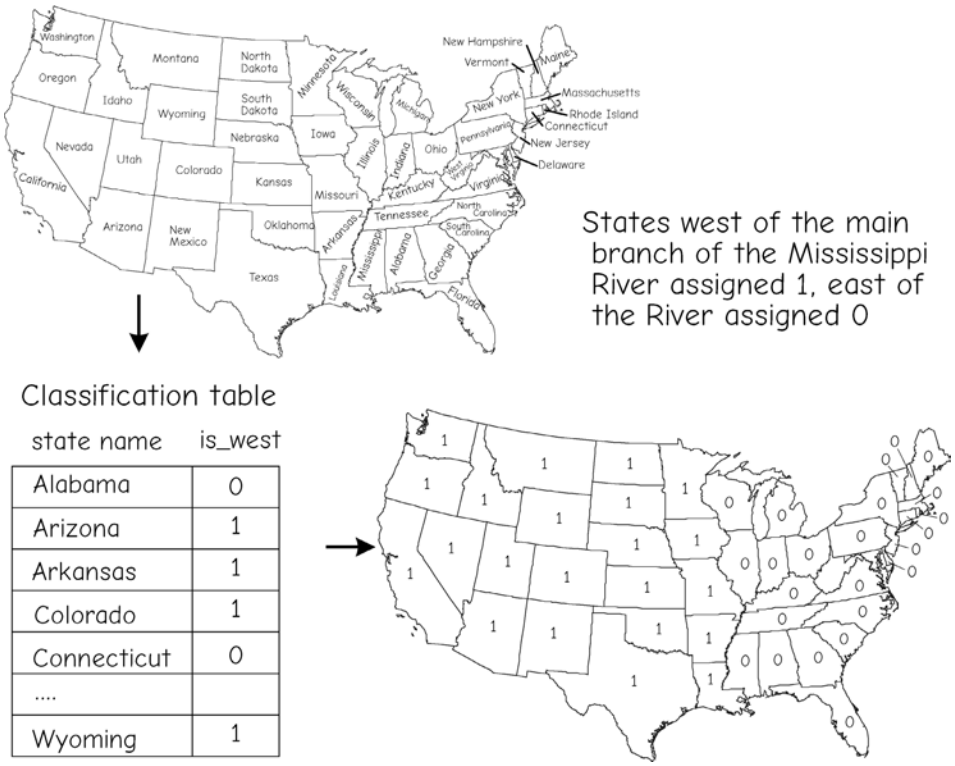


Figure 9-14: The classification of a thematic layer. Values are given to specific attributes in a classification table, which is used, in turn, to assign classes in an output layer.

This target attribute is assigned a unique value. The target attribute is assigned a different value for all unselected features. This creates a column that identifies the selected set; for example, all counties that are small, but with a large population.

For example, we may wish to select states at least partially west of the Mississippi River as an intermediate step in an analysis (Figure 9-15). We may be using this classification in many subsequent spatial operations. Thus, we wish to store this characteristic, whether the state is west or east of the Mississippi River. States are selected based on location and reclassified. We record this classification by creating a new attribute and assigning a binary value to this attribute, 1 for those parcels that satisfy the criteria, and 0 for those that do not (Figure 9-15). The variable is_west records the state location relative to the Mississippi River. Additional selection operations may be applied, and the created binary variable preserves the information generated in the initial selection.

Previous examples have shown vector data, but we may also reclassify raster data. If the input raster values are nominal or ordinal data, the reclassification will look very similar to the vector examples shown in Figure 9-14. A list of input and corresponding outputs are provided, and the reclassification operates on a cell-by-cell basis. When interval/ratio raster data are used as input, then input ranges are required rather than specific input values. This distinction is described in detail in Chapter 10.



state name	is_west
Alabama	0
Arizona	1
Arkansas	1
Colorado	1
Connecticut	0
....	
Wyoming	1

Figure 9-15: An example of a binary classification. Features are placed into two classes in a binary classification, west (1) and east (0) of the Mississippi River. The classification table codifies the assignment.

Data-defined Classification

Manually defining the classification table may not always be necessary, and may be tedious or complex. Suppose we wish to assign a set of display colors to a set of elevation values. There may be thousands of distinct elevation values in the data layer, and it would be inconvenient at best to assign each color manually. Data-defined classification methods, where class intervals are automatically derived from rules applied to the input data, are often used in these instances.

An automatic classification uses some rule to specify the input class to output class assignments. The input and output class boundaries are often based on a set of parameters used to guide class definition.

A potential drawback from an automated class assignment stems from our inability to precisely specify class boundaries. A mathematical formula or algorithm defines the class boundaries, and so specific classes of interest may be split. Thus, the analyst sacrifices precise control over class specification when an automated classification is used.

Figure 9-16 describes a data layer we will use to illustrate automatic class assignment. The figure shows a set of "neighborhoods" with populations that range from 0 to 5133. We wish to display the neighborhoods and populations in three distinct classes, high, medium, and low population. High will be shown in black, medium in gray, and low in white. We must decide how to assign the categories — what population levels define high, medium, and low? In many applications, the classification levels are previously defined. There may be an agreed-upon standard for high population, and we would simply use this level. However, in many instances the classes are not defined, and we must choose them.

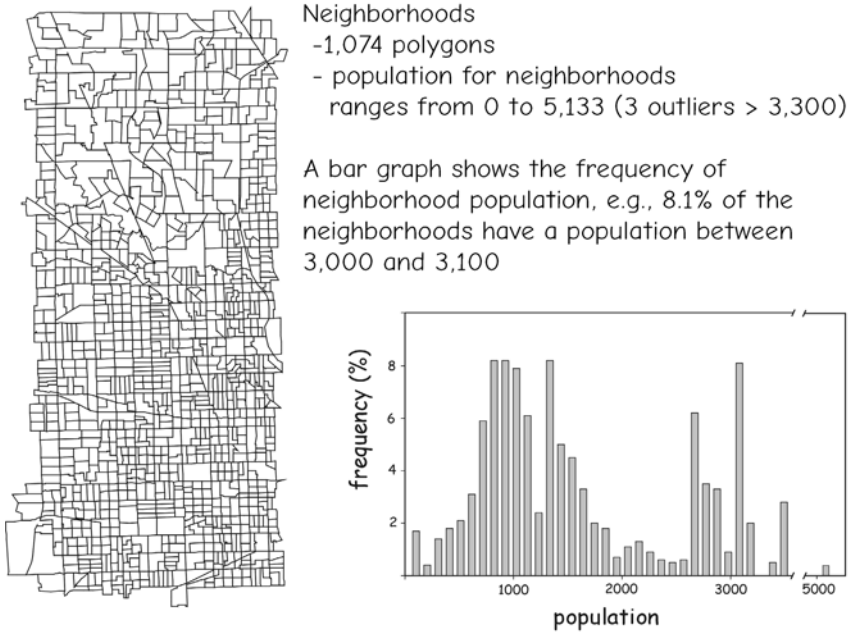


Figure 9-16: Neighborhood polygons and population levels used in subsequent examples of classification assignment. The populations for these 1,074 neighborhoods range from 0 to 5,133. The histogram at the lower right shows the frequency distribution. Note that there is a break in the chart between 3,500 and

Figure 9-16 includes a bar graph depicting the population frequency distribution; this type of bar graph is commonly called a histogram. The frequency histogram shows the number of neighborhoods that are found in each bar (or "bin") of a set of very narrow population categories. For example, we may count the number of neighborhoods that have a population between 3,000 to 3,100. Approximately 8.1% of the neighborhoods have a population in this range, so a vertical bar corresponding to 8.1 units high is plotted. We count and plot the histogram values for each of our narrow categories (e.g., the number from 0 to 100, from 100 to 200, from 200 to 300), until the highest population value is plotted.

Our primary decision in class assignment is where to place the class boundaries. Should we place the boundary between the low and medium population classes at 1,000, or at 1,200? Where should the boundary between medium and high population classes be placed? The location of the class boundaries will change the appearance of the map, and also the resulting classification.

One common method for automatic classification specifies the number of output classes and requests equal-interval classes over the range of input values. This *equal-interval* classification simply subtracts the lowest value of the classification variable from the highest value, and defines equal-width boundaries to fit the desired number of classes into the range.

Figure 9-17 illustrates an equal-interval classification for the population variable. Three classes assigned over the range of 0 to 5,133 are specified. Each interval is approximately one-third of this range. This range is evenly divided by 1,711. The small

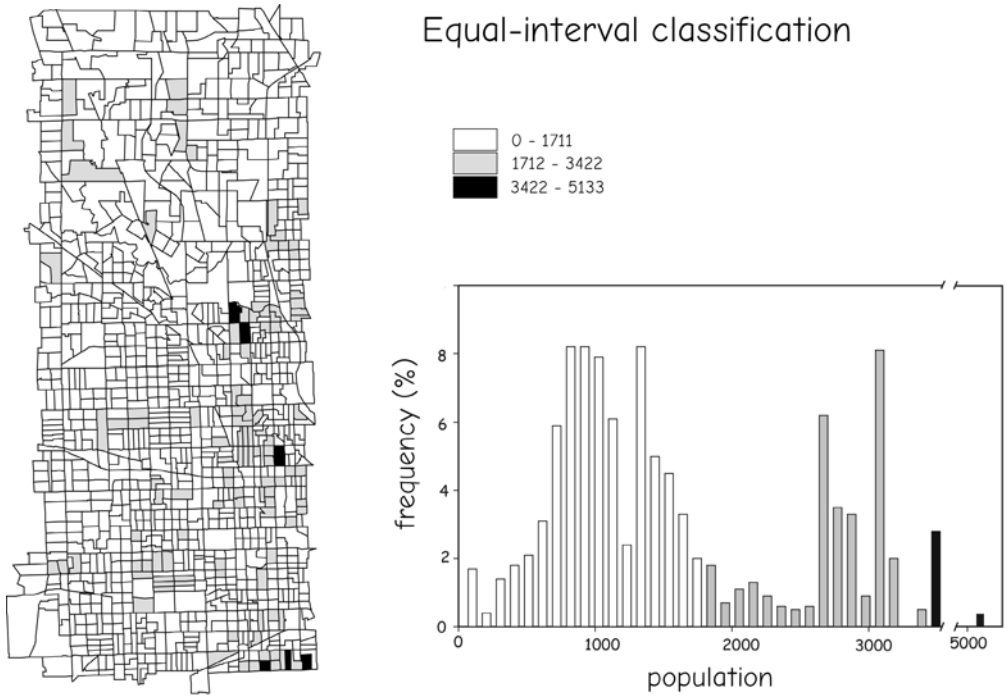


Figure 9-17: An equal-interval classification. The range 0 to 5,133 is split into three equal parts. Colors are assigned as shown in the map of the layer (left), and in the frequency plot (right). Note the relatively few polygons assigned to the high classes in black. A few neighborhoods with populations near 5,000 shift the class boundaries upward.

class extends from 0 to 1,711, the medium class from 1,712 to 3,422, and the large class from 3,423 to 5,133. Population categories are shown colored accordingly on the map and the bar graph, with the small (white), medium (gray), and large (black) classes shown.

Note that the low population class shown in white dominates the map; most of the neighborhoods fall in this population class. This often happens when there are features that have values much higher than the norm. There are a few neighborhoods with populations above 5,000 (to the right of the break in the population axis of the bar graph), while most neighborhoods have populations below 3,000. The outliers shift the class boundaries to higher values, 1711 and 3,422, resulting in most neighborhoods falling in the small population category.

Another common method for class assignment results in an *equal-area* classification (Figure 9-18). Class boundaries are defined to place an equal proportion of the study area into each of a specified number of classes. This usually leads to a visually balanced map because all classes have approximately equal extents. Equal-area classes are often desirable, for example, when resources need to be distributed over equal areas, or when equally sized overlapping sales territories may be specified.

Class width may change considerably with an equal-area classification. An equal-area classification sets class boundaries so that each class covers approximately the same area. A class may consist of a few or even one large polygon. This results in a small range for the large polygon classes. Classes also tend to have a narrow range of values near the peaks in the histogram. Many polygons are represented at the his-

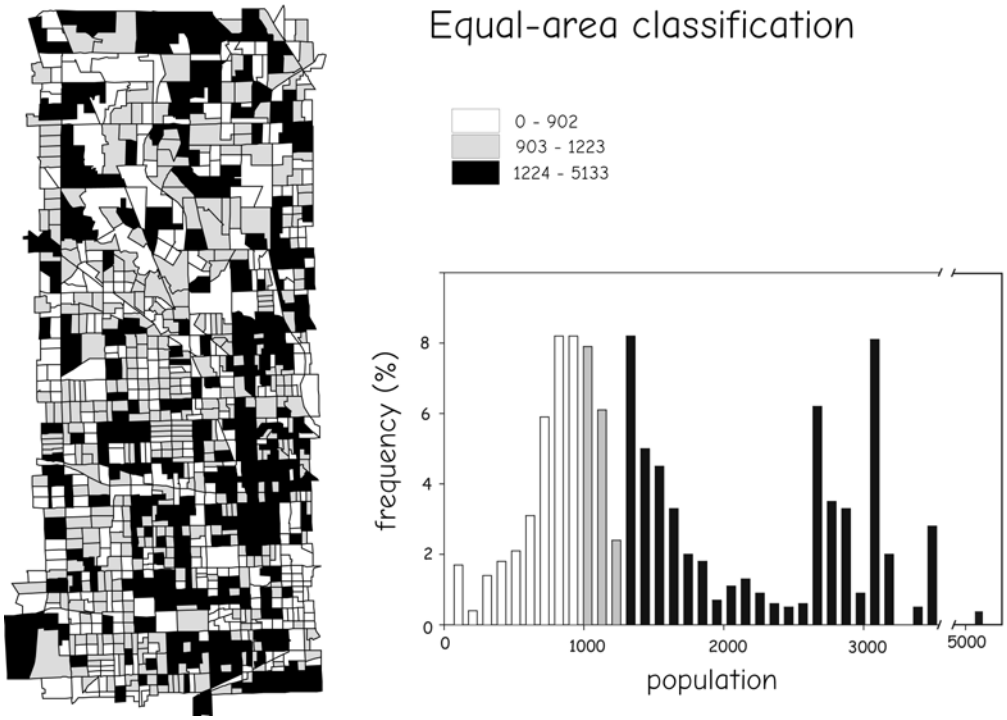


Figure 9-18: Equal-area classification. Class boundaries are set such that each class has approximately the same total area. This often leads to a smaller range when groups of frequent classes are found. In this instance the medium class spans a small range, from 903 to 1,223, while the high population class spans a range that is almost 10 times broader, from 1,224 to 5,133.

togram peaks, and so these may correspond to large areas. Both of these effects are illustrated in Figure 9-18. The middle class of the equal-area classification occurs at population values between 903 and 1223. This range of populations is near the peak in the frequency histogram, and these population levels are associated with larger polygons. This middle class spans a range of approximately 300 population units, while the small and large classes span near 900 and 4,000 population units, respectively.

Equal-area assignments may be highly skewed when there are a few polygons with large areas, and these polygons have similar values. Although not occurring in our example, there may be a relationship between the population and area for a few neighborhoods. Suppose in a data set similar to ours there is one very large neighborhood dominated by large parks. This neighborhood has both the lowest populations and largest area. An equal-area classification may place this neighborhood in its own class. If a large parcel also occurs with high population levels, we may get three classes: one parcel in the small class, one parcel in the high population class, and all the remaining parcels in the medium population class. While most equal-area classifications are not this extreme, unique parcels may strongly affect class ranges in an equal-area classification.

We will cover a final method for automated classification, a method based on *natural breaks*, or gaps, in the data (Figure 9-19). Natural breaks classification looks for "obvious" breaks. It attempts to identify naturally occurring clusters of data; not clusters based on the spatial relationships, but rather clusters based on an ordering variable.

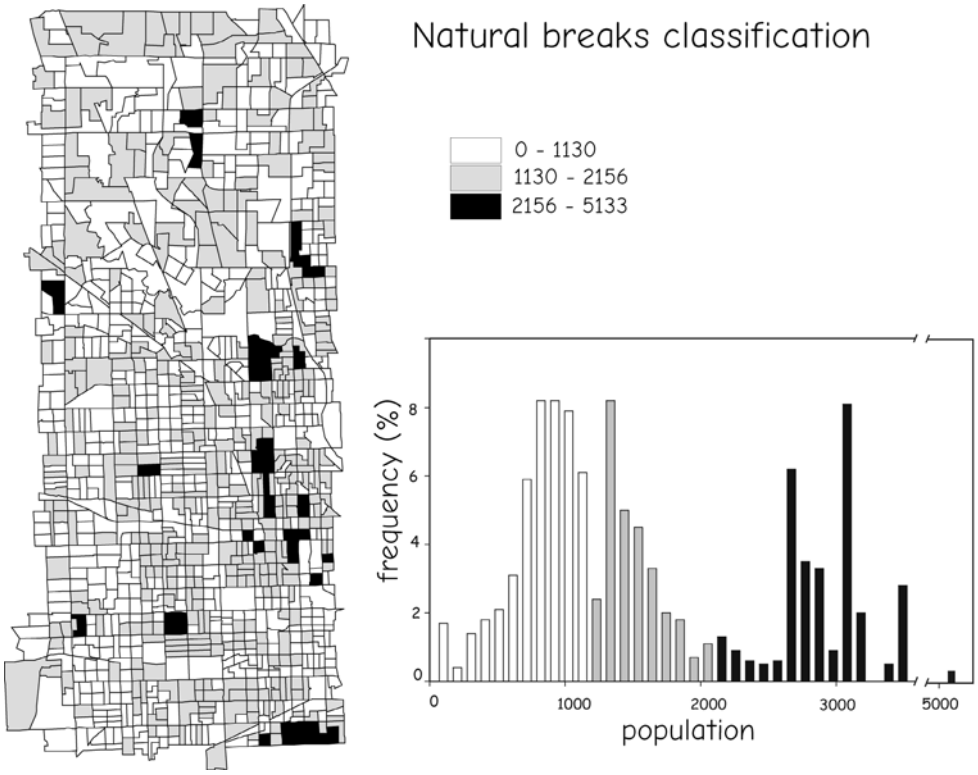


Figure 9-19: Natural breaks classification. Boundaries between classes are placed where natural gaps or low points occur in the frequency distribution.

There are various methods used to identify natural breaks. Large gaps in an ordered list of values are one common method. The values are listed from lowest to highest, and the largest gaps in values selected. Barring gaps, low points in the frequency histogram may be identified. There is usually an effort to balance the need for relatively wide and evenly distributed classes and the search for natural gaps. Many narrow classes and one large class may not be acceptable in many instances, and there may be cases where the specified number of gaps does not occur in the data histogram. More classes may be requested than obvious gaps, so some natural break methods include an alternative method, for example, equally spaced intervals, for portions of the histogram where no natural gaps occur.

Figure 9-19 illustrates a natural break classification. Two breaks are evident in the histogram, one near 1300 and one near 2200 persons per neighborhood. Small, medium, and large populations are assigned at these junctures.

Figure 9-17 through Figure 9-19 illustrate an important point: you must be careful when interpreting class maps, because the apparent relative importance of categories may be manipulated by altering the starting and ending values in each class. Figure 9-17 suggests most neighborhoods are low population, Figure 9-18 suggests that high population neighborhoods cover the largest areas and that they are well mixed with areas of lower population, while Figure 9-19 indicates the area is dominated by low and medium population neighborhoods. Precisely because there are no objectively defined population boundaries, we have great flexibility in manipulating the impression we create. The legend in class maps should be scrutinized, and the range between class boundaries noted. .

The Modifiable Areal Unit Problem

When there are no objectively recognizable categories, polygons may be reclassified and grouped in many ways.The aggregate values for the classified polygons, such as class population, age, and income, will depend on the size, shape, and location of the aggregated polygons (Figure 9-20). This general phenomenon, known as the modifiable areal unit problem or MAUP, has been exploited by politicians to redraw political boundaries to one party or another's — but generally not the country's — advantage. The process of aggregating neighborhoods to create majority blocks for political advantage was named gerrymandering after Massachusetts governor Elbridge Gerry, when he crafted a political district shaped like a salamander.

There are two primary characteristics of the MAUP that may be manipulated to affect aggregate polygon values. The first is the *zoning effect*, that aggregate statistics may change by the shape of the units, and the second is the *size effect*, that aggregate statistics may change with the area of the units. For example, the mean income of a unit will change when the boundaries of a unit change, either because of a change in zone or a change in size.

MAUP effects may substantially influence the values for each unit, and hence subsequent analysis. Openshaw and Taylor published results in 1979 that illustrate MAUP dependencies particularly well. They analyzed the percentage of elderly voters and the number of Republican voters for the 99 counties in Iowa. They showed that the correlation between the elderly voters and Republican voter numbers ranged from 0.98 to -0.81 by varying the scale and aggregation units that grouped counties. Additional work has shown that multivariate statistical models based on aggregate data are similarly dependent on the aggregation units, leading to contradictory results predictions.

Numerous studies of the MAUP have shown how to identify and/or reduce the pri-

mary negative impacts of the zoning and size effects. The primary recommendation is to work with the basic units of measure. In our census example, this would be to collect and maintain information on the individual person or household. This is often not possible; for example, aggregation is specifically required to maintain the anonymity of the census respondents. However, many efforts allow recording and maintaining data on the primary units within a GIS framework, and this should be implemented when possible.

A second way to address the MAUP is based on optimal zoning. Zones are designed to maximize variation between zones while minimizing variation within zones. Optimal zones are difficult to define for more than one variable, because variables often do not change in concert. For example, an optimal set of zones for determining traffic densities may not be an optimal for average age. Old people are no more nor less likely to live near busy or low traffic roads. Optimum zoning approaches are best applied when interest in one variable predominates.

Another approach to solving the MAUP involves conducting a set of sensitivity analyses. Units are aggregated and rezoned across a range of sizes and shapes and the analyses performed for each set. Changes in the results are observed, and the sensitivity to zone boundaries and sizes noted. These tests may identify the relative sensitivities of different variables to size and zoning effects. Robust results may be identified, for example, average age may not change over a range of sizes and unit combinations, yet may change substantially over a narrow range of sizes in some areas. This approach requires many computations, because it uses replicated runs for each set of variables, zone levels, and shapes. This often overwhelms the available computing resources for many problems and agencies.

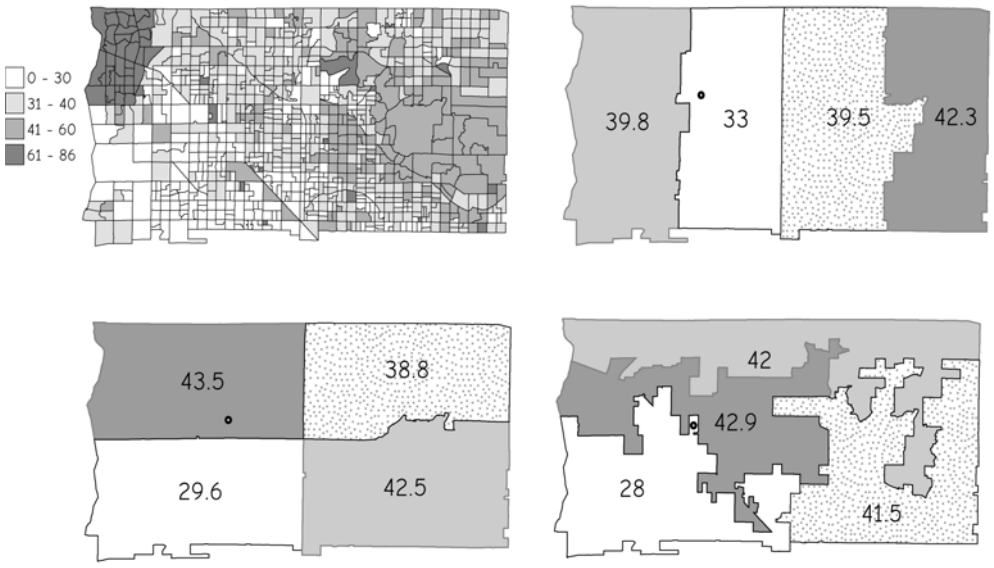


Figure 9-20: An example of the modifiable areal unit problem (MAUP). Census blocks (upper left) have been aggregated in various ways to produce units that show different mean population age. All units are approximately equal in size. Note that the number of zones with a mean age over 40 can be one (upper right), two (lower left), or three (lower right), and that an individual block (small circle, upper left quadrant) may be in a polygon with a mean age in the 20s, 30s, or 40s, depending on unit shape.

Dissolve

A *dissolve* function is primarily used to combine similar features within a data layer. Adjacent polygons may have identical values for an attribute. For example, a wetlands data layer may specify polygons with several subclasses, such as wooded wetlands, herbaceous wetlands, or open water. If an analysis requires we identify only the wetland areas vs. the upland areas, then we may wish to dissolve all boundaries between adjacent wetlands. We are only interested in preserving the wetland/upland boundaries.

Dissolve operations are usually applied based on a specific "dissolve" attribute associated with each feature. A value or set of values is identified that belongs in the same grouping. Each line that serves as a boundary between two polygon features is assessed. The values for the dissolve attribute are compared across the boundary line. If the values are the same, the boundary line is removed, or dissolved away. If the values for the dissolve attribute differ across the boundary, the boundary line is left intact.

Figure 9-21 illustrates the dissolve operation that produces a binary classification. This classification places each state of the contiguous United States into one of two categories, those entirely west of the Mississippi River (1) and those east of the Mississippi River (0). The attribute named is_west contains values indicating location. A dissolve operation applied on the variable is_west removes all state boundaries between similar states. This reduces the set from 48 polygons to two polygons.

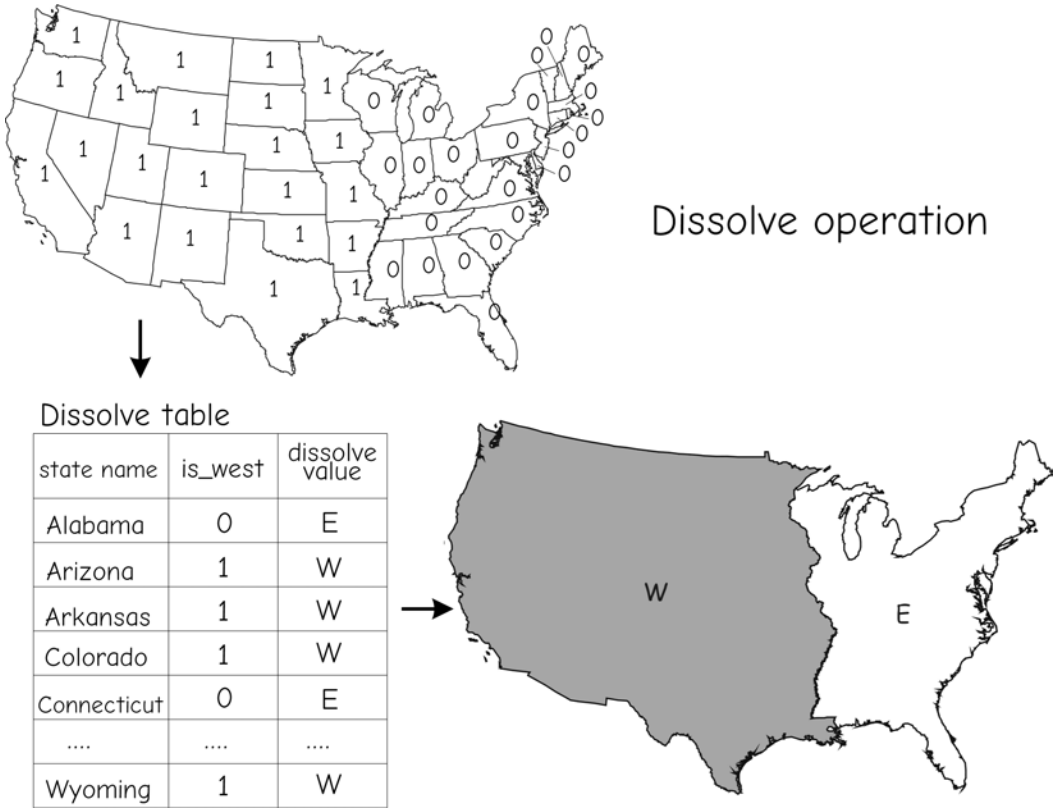


Figure 9-21: An illustration of a dissolve operation. Boundaries are removed when they separate states with the same value for the dissolve attribute **is_west**.

Dissolve operations are often needed prior to applying an area-based selection in spatial analysis. For example, we may wish to select areas from the natural breaks classification shown on the left of Figure 9-22. We seek polygons that are greater than 3 mi^2 in area and have a medium population. The polygons may be composed of multiple neighborhoods. We typically must dissolve the boundaries between adjacent, medium-sized neighborhoods prior to applying the size test. Otherwise, two adjacent, medium population neighborhoods may be discarded because both cover approximately 2 mi^2. Their total area is 4 mi^2, above the specified threshold, yet they will not be selected unless a dissolve is applied first.

Dissolves are also helpful in removing unneeded information. After the classification into small, medium, and large size classes, many boundaries may become redundant. Unneeded boundaries may inflate storage and slow processing. A dissolve has the advantage of removing unneeded geographic and tabular data, thereby simplifying data, improving processing speed, and reducing data volumes.

Figure 9-22 illustrates the space saving and complexity reduction common when applying a dissolve function. The number of polygons is reduced approximately ninefold by the dissolve, from 1,074 on the left to 119 polygons on the right of Figure 9-22.

Figure 9-22: An example of a dissolve operation. Note the removal of lines separating polygons of the same size class. This greatly reduces the number of polygons.

Attribute Aggregation in a Dissolve Operation

We often wish to transfer information in the attribute table when applying a dissolve function. For example, we may wish to find the total population of a set of neighborhoods that are within walking distance of a set of bus stops. Each neighborhood polygon contains a population count attribute, and we wish to sum the values across the neighborhoods that correspond to each bus stop. The new dissolved polygons, representing the neighborhoods closest to each bus stop, will contain a summed population variable. We might then do further analysis to identify areas where new bus stops might be needed, or to recommend a change in bus frequency.

Aggregation functions allow us to preserve information in a dissolve operation. Typically we may sum, average, calculate the range, maximum, minimum, or other common statistics, and assign these to the output polygons. The functions first identify the adjacent polygons that will be combined to form the new polygons, and then apply the specified operation to target attribute variables. Figure 9-23 diagrams a dissolve that sums cost across input polygons. Adjacent polygons of the same type are combined, and the values of the component polygons summed. Different analyses might require different aggregation statistics, e.g., average, maximum, or range (Figure 9-24).

Some of the variables might be nonsensical as inputs or as outputs, so care must be taken when aggregating during a dissolve. This is particularly true for area averaged values, or for categorical or ordi-

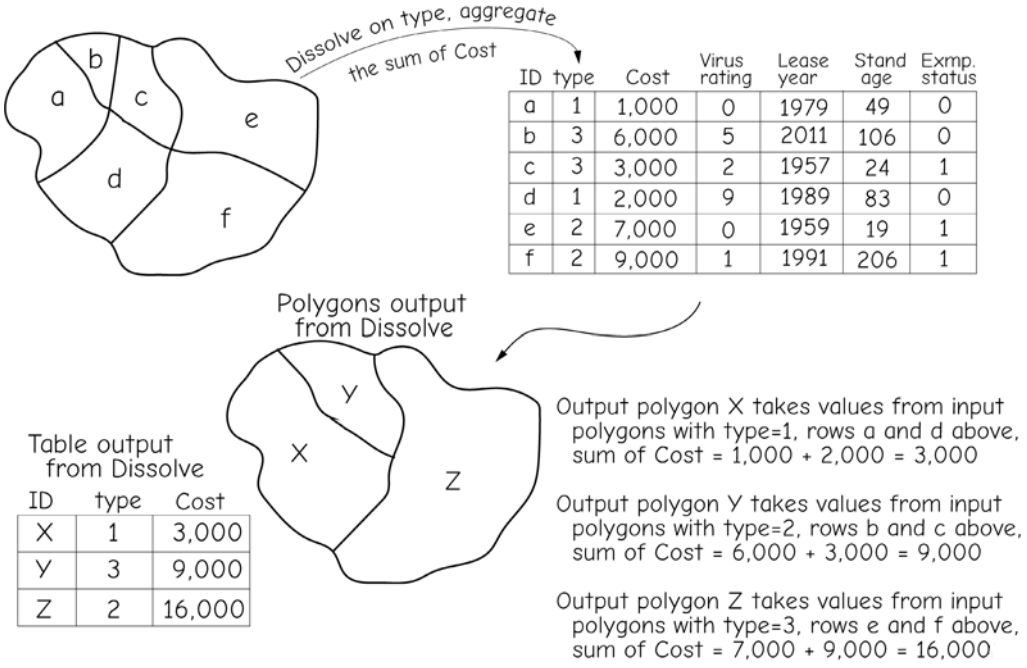


Figure 9-23: An example aggregation during a dissolve operation. Adjacent polygons of the same type are combined. For each combination, the input polygons costs are summed and this sum assigned to a cost variable for the aggregated polygon in the output data layer.

nal values as inputs. If I average the average population of two input polygons, I will get an erroneous average for the output polygon, except in the rare case when both polygons have the same area. Proof of this is left to the reader. Aggregates of categorical or ordinal variables also require caution, as the average or sum of ordinal or category values has meaning in relatively few applications.

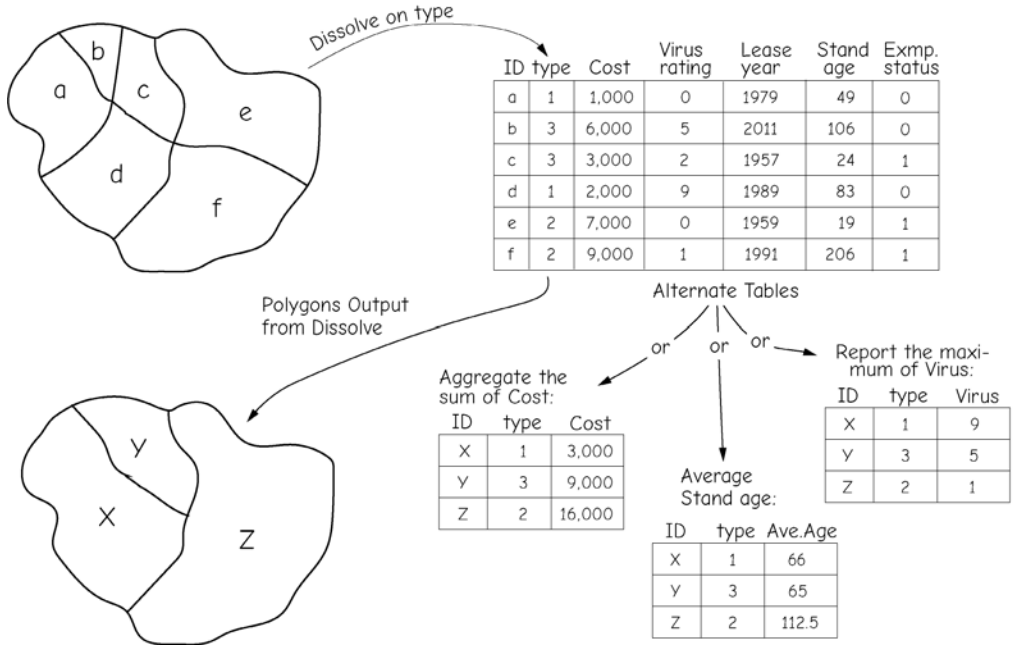


Figure 9-24: Different aggregation operations may be applied, and appropriate, for different input variables. Here are examples of sum, average, and maximum operations applied during a dissolve operation.

Proximity Functions and Buffering

Proximity functions or *operations* are among the most powerful and common spatial analysis tools. Many important questions hinge on proximity, the distance between features of interest. How close are schools to an oil refinery, what neighborhoods are far from convenience stores, and which homes will be affected by an increase in freeway noise? Many questions regarding proximity are answered through spatial analyses in a GIS. Here we focus on proximity functions that create new features and layers, rather than proximity selection, described earlier.

Proximity functions modify existing features or create new features that depend in some way on distance. For example, one simple proximity function creates a raster of the minimum distance from a set of features (Figure 9-25). The figure shows a distance function applied to water holes in a wildlife reserve. Water is a crucial resource for nearly all animals, and the reserve managers may wish to ensure that most of the area is within a short distance of water. In this instance point features are entered that represent the location of permanent water. Water holes are represented by individual points, and rivers by a group of points set along the river course. A proximity function calculates the distance to all water points for each raster cell. The minimum distance is selected and placed in an output raster data layer (Figure 9-25). The distance function creates a mosaic of what appear to be overlapping circles. Although the shading scheme shows apparently abrupt transitions, the raster cells contain a smooth gradient in distance away from each water feature.

Distance values are most often calculated based on the Pythagorean formula (Figure 9-26). These values are typically calculated from cell center to cell center when applied to a raster data set. Although any distance is possible, the distances between adjacent cells change in discrete intervals related to the cell size. Note that distances are not restricted to even multiples of the cell size, because distances measured on diagonal angles are not even multiples of the cell dimension. There may

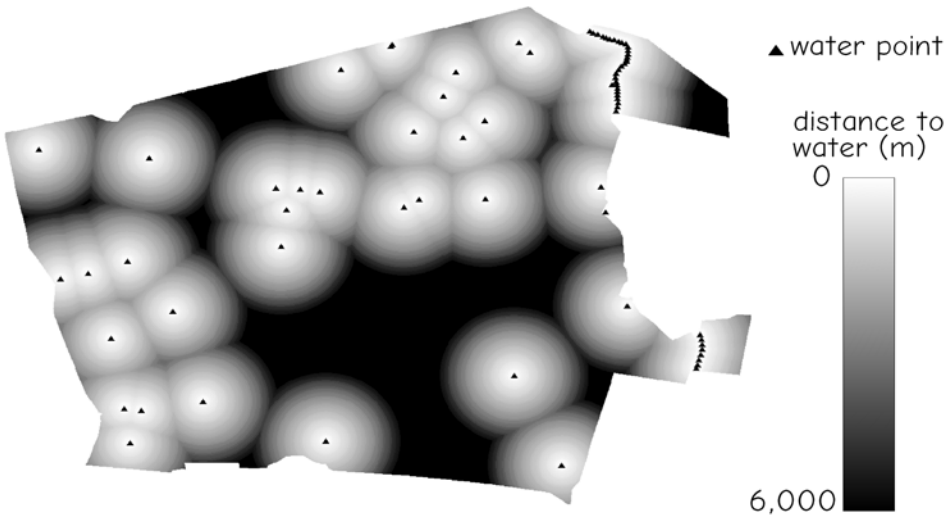


Figure 9-25: An example of a distance function. This distance function is applied to a point data layer and creates a raster data layer. The raster layer contains the distance to the nearest water feature.

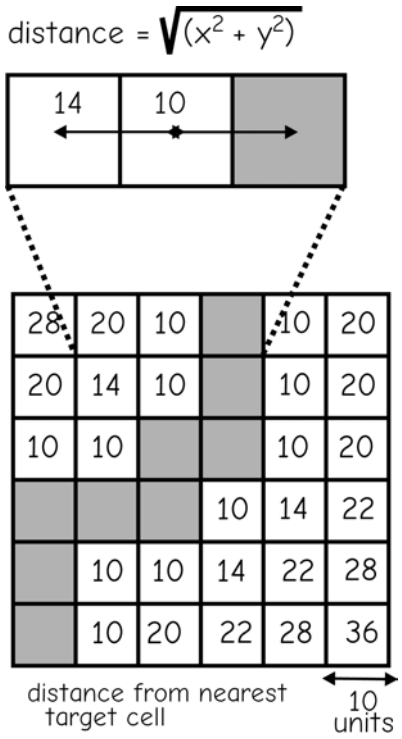


Figure 9-26: A distance function applied to a raster data set.

be no cells that are exactly some fixed distance away from the target features; however, there may be many cells less than or greater than that fixed distance.

Buffers

Buffering is one of the most commonly used proximity functions. A *buffer* is a region that is less than or equal to a specified distance from one or more features (Figure 9-27). Buffers may be determined for point, line, or area features, and for raster or vector data. Buffering is the process of creating buffers. Buffers typically identify areas that are "outside" some given threshold distance compared to those "inside" some threshold distance.

Buffers are used often because many spatial analyses are concerned with distance constraints. For example, emergency planners might wish to know which schools are within 1.5 kilometers of an earthquake fault, a park planner may wish to identify all lands more than 10 kilometers from the nearest highway, or a business owner may wish to identify all potential customers within a given radius of her store. All these questions may be answered with the appropriate use of buffering.

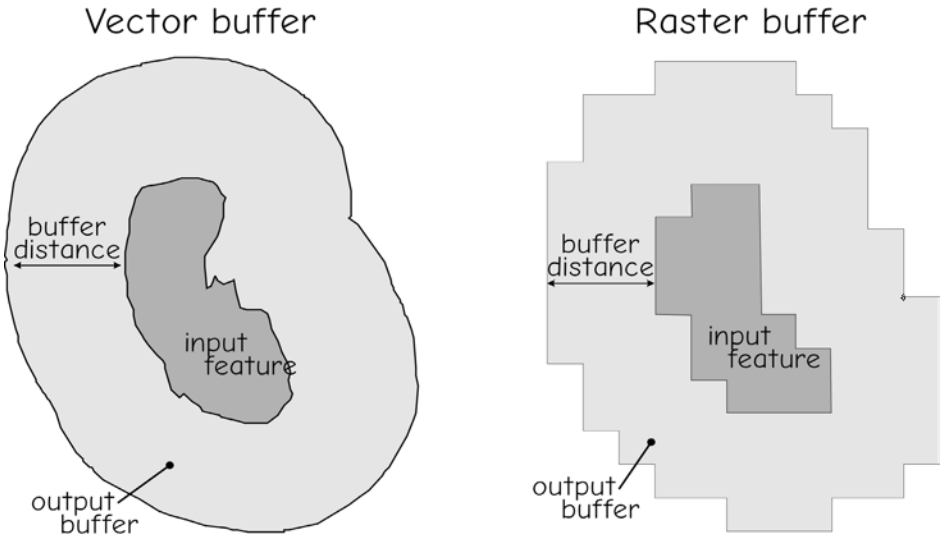


Figure 9-27: Examples of vector and raster buffers derived from polygonal features. A buffer is defined by those areas that are within some buffer distance from the input features.

Raster Buffers

Buffer operations on raster data entail calculating the distance from each source cell center to all other cell centers. Output cells are assigned an in value whenever the cell-to-cell distance is less than the specified buffer distance. Those cells that are further than the buffer distance are assigned an out value (Figure 9-28).

Raster buffers combine a minimum distance function and a binary classification function. A minimum distance function calculates the shortest distance from a set of target features and stores this distance in a raster data layer. The binary classification function splits the raster cells into two classes: those with a distance greater than the threshold value, and those with a distance less than or equal to a threshold value.

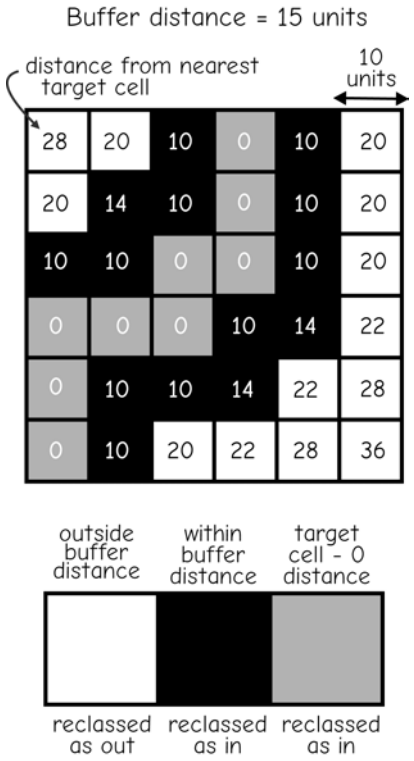


Figure 9-28: Raster buffering as a combination of distance and classification. Here, cells less than 15 units from the target cells are identified.

Buffering with raster data may produce a "stair-step" boundary, because the distance from features is measured between cell centers. When the buffer distance runs parallel and near a set of cell boundaries, the buffer boundary may "jump" from one row of cells to the next (Figure 9-28). This phenomenon is most often a problem when the raster cell size is large relative to the buffer distance. A buffer distance of 100 m may be approximated when applied to a raster with a cell size of 30 m. A smaller cell size relative to the buffer distance results in less obvious "stair-stepping." The cell size should be small relative to the spatial accuracy of the data, and small relative to the buffer distance. If this rule is followed, then stair-stepping should not be a problem, because buffer sizes should be many times greater than the uncertainty inherent in the data.

Vector Buffers

Vector buffering may be applied to point, line, or area features, but regardless of input, buffering always produces an output set of area features (Figure 9-29). There are many variations in vector buffering. *Simple buffering*, also known as *fixed distance buffering*, is the most common form of vector buffering (Figure 9-29). Simple buffering identifies areas that are a fixed distance or greater from a set of input features. Simple buffering does not distinguish between regions that are close to one feature from those that are close to more than one feature. A location is either within a given distance from any one of a set of features, or farther away.

Simple buffering uses a uniform buffer distance for all features. A buffer distance of 100 meters specified for a roads layer may be applied to every road in the layer, irrespective of road size, shape, or location. In a similar manner, buffer distances for all points in a point layer will be uniform, and buffer distances for all area features will be fixed.

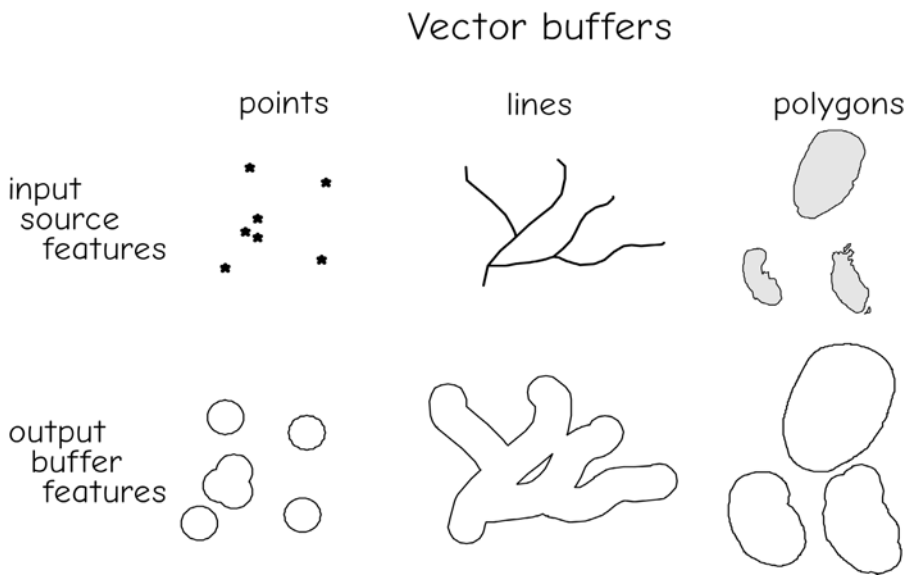


Figure 9-29: Vector buffers produced from point, line, or polygon input features. In all cases, the output is a set of polygon features.

Buffering on vector point data is based on the creation of circles around each point in a data set. The equation for a circle with an origin at $x=0$, $y=0$ is:

$$r = \sqrt{x^2 + y^2} \qquad (9.1)$$

where r is the buffer distance. The more general equation for a circle with a center at x_1, y_1, is:

$$r = \sqrt{(x - x_1)^2 + (y - y_1)^2} \qquad (9.2)$$

Equation (9.2) reduces to equation (9.1) at the origin, where $x1 = 0$, and $y1 = 0$. The general equation creates a circle centered on the coordinates $x1$, $y1$, with a buffer distance equal to the radius, r. Point buffers are created by applying this circle equation successively to each point feature in a data layer. The x and y coordinate locations of each point feature are used for x_1 and y_1, placing the point feature at the center of a circle.

Buffered circles may overlap, and in simple buffering, the circle boundaries that occur in overlap areas are removed. For example, areas within 10 km of hazardous waste sites may be identified by creating a buffer layer. We may have a data layer in which hazardous waste sites are represented as points (Figure 9-30a). A circle with a 10 km radius is drawn around each point. When two or more circles overlap, internal boundaries are dissolved, resulting in noncircular polygons (Figure 9-30b).

More complex buffering methods may be applied. These methods may identify buffer areas by the number of features within the given buffer distance, or apply variable buffer distances depending on the characteristics of the input features. We may be interested in areas that are near multiple hazardous waste sites. These areas near multiple hazardous sites may entail added risk and therefore require special monitoring or treatment. We may be mandated to identify all areas within a buffer

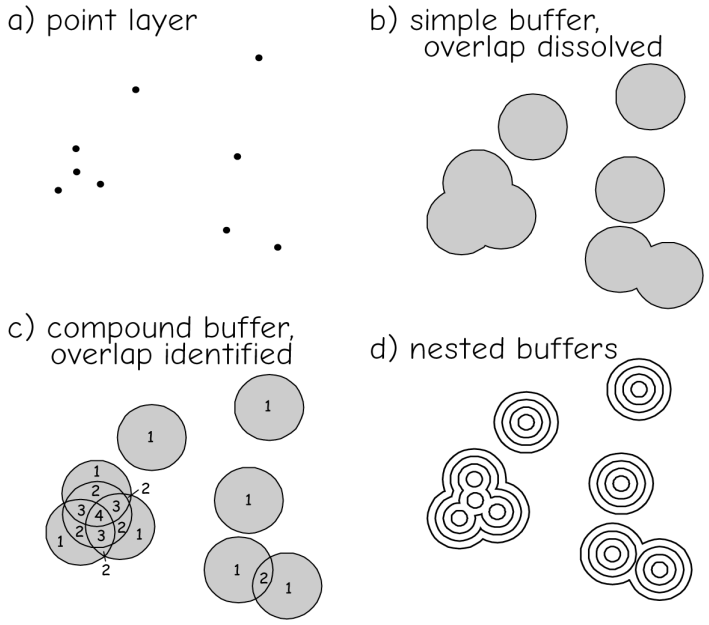


Figure 9-30: Various types of point buffers. Simple buffers dissolve areas near multiple features, more complex buffers do not. Multiring buffers provide distance-defined zones around each feature.

distance of a hazardous waste site, and the number of sites. In most applications, most of the dangerous areas will be close to one hazardous waste site, but some will be close to two, three, or more sites. The simple buffer, described above, will not provide the required information.

A buffering variant, referred to here as *compound buffering*, provides the needed information. Compound buffers maintain all overlapping boundaries (Figure 9-30c). All circles defined by the fixed radius buffer distance are generated. These circles are then intersected to form a planar graph. For each area, an attribute is created that records the number of features within the specified buffer distance.

Nested (or multiring) buffering is another common buffering variant (Figure 9-30d). We may require buffers at multiple distances. In our hazardous waste site example, suppose threshold levels have been established with various actions required for each threshold. Areas very close to hazardous waste sites require evacuation, intermediate distance require remediation, and areas farther away require monitoring. These zones may be defined by nested buffers.

Buffering on vector line and polygon data is also quite common. The formation of line buffers may be envisioned as a sequence of steps. First, circles are created that are centered at each node or vertex (Figure 9-31). Tangent lines are then generated. These lines are parallel to the input feature lines and tangent to the circles that are centered at each node or vertex. The tangent lines and circles are joined and interior circle segments dissolved.

Variable distance buffers (Figure 9-32) are another common variant of vector buffering. As indicated by the name, the buffer distance is variable, and may change among features. The buffer distance may increase in steps; for example, we may have one buffer distance for a given set of features, and a different buffer distance for the remaining features. In contrast, the buf-

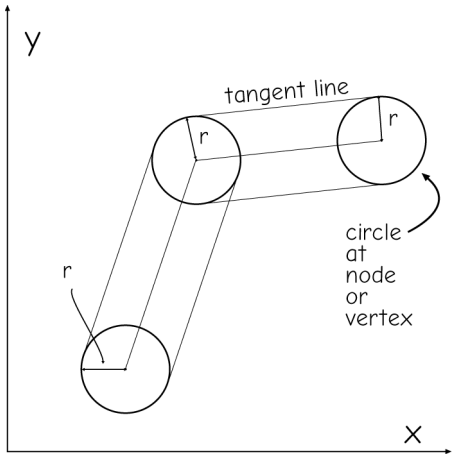


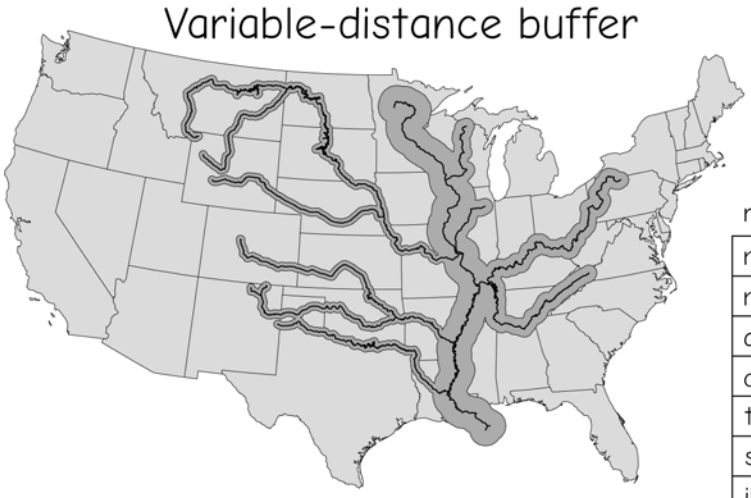
Figure 9-31: The creation of a line buffer at a fixed distance r.

fer distance may vary smoothly; for example, the buffer distance around a city may be a function of the population density in the city.

There are many instances for which we may require a variable distance buffer. We may wish to specify a larger buffer zone around large fuel storage facilities when compared to smaller fuel storage facilities. We often require more stringent protections further away from large rivers than for small rivers, and give large landfills a wider berth than small landfills.

Figure 9-32 illustrates the creation of buffers around a river network, where distance varies by river size. The increase in distance may be motivated by an increased likelihood of flooding downstream or an increased sensitivity to pollution. We may specify a buffer distance of 50 km for small rivers, 75 km for intermediate size rivers, and 100 km for large rivers. There are many other instances when variable distance buffers are required, for example, road noise, smoke stacks, or landfills.

The variable buffer distance is often specified by an attribute in the input data layer (Figure 9-32). A portion of the attribute table for the river data layer is shown. The attribute table contains the river name in river_identifier and the buffer distance is stored in buffdist. The attribute buffdist is accessed during buffer creation, and the size of the buffer adjusted automatically for each line segment. Note how the buffer size depends on the value in buffdist.



river_identifier	buffdist
mississippi	100
missouri	50
arkansas	50
ohio	75
tennessee	75
st. croix	75
illinois	75
wisconsin	75

Figure 9-32: An illustration of a variable distance buffer. A line buffer is shown with a variable buffer distance based on a river_identifier. A variable buffer distance, buffdist, is specified in a table and applied for each river segment.

Overlay

Overlay operations are powerful spatial analysis tools, and were an important driving force behind the development of GIS technologies. Overlays involve combining spatial and attribute data from two or more spatial data layers, and they are among the most common and powerful spatial data operations (Figure 9-33). Many problems require the overlay of thematically different data. For example, we may wish to know where there are inexpensive houses in good school districts, where whale feeding grounds overlap with proposed oil drilling areas, or the location of farm fields that are on highly erodible soils. In the latter example, a soils data layer may be used to identify highly erodible soils, and a current land use layer may be used to identify the locations of farm fields. The boundaries of erodible soils will not coincide with the boundaries of the farm fields in most instances, so these soils and land use data must somehow be combined. Overlay is the primary means of providing this combination.

An overlay operation requires that data layers use a common coordinate system. Overlay uses the coordinates that define each spatial feature to combine the data from the input data layers. The coordinates for any point on the Earth depend on the coordinate system used (Chapter 3). If the coordinate systems used in the various layers are not exactly the same, the features in the data layers will not align correctly.

Overlay may be viewed as the vertical stacking and merger of spatial data (Figure 9-33). Features in each data layer are set one "on top" another, and the points, lines, or area feature boundaries are merged into a single data layer. The attribute data are also combined so that the new data layer includes the information contained in each input data layer.

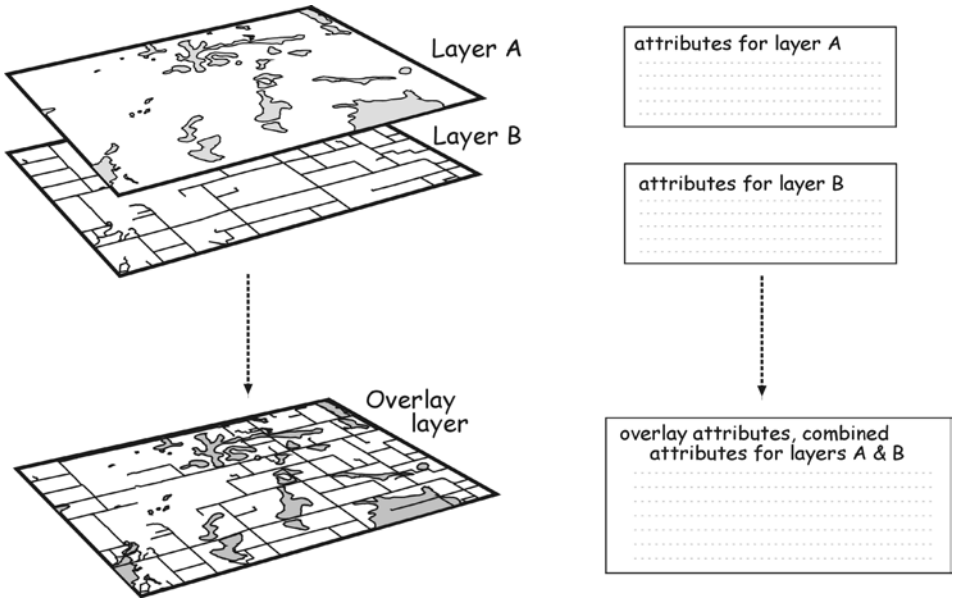


Figure 9-33: Spatial data overlay. Overlay combines both the coordinate information and the attribute information across different data layers.

Vector Overlay

Overlay when using a vector data model involves combining the point, line, and polygon geometry and associated attribute data. This overlay creates new geometry. Overlay involves the merger of both the coordinate and attribute data from two vector layers into a new data layer. The coordinate merger may require the intersection and splitting of lines or areas and the creation of new features.

Figure 9-34 illustrates the overlay of two vector polygon data layers. This overlay requires the intersection of polygon boundaries to create new polygons. The overlay combines attribute data during polygon overlay. The data layer on the left is composed of two polygons. There are only two attributes for `Layer 1`, one an identifier (`ID`), and the other specifying values for a variable named `class`. The second input data layer, `Layer 2`, also contains two polygons, and two attributes, `ID` and `cost`. Note that the two tables have an attribute with the same name, `ID`. These two `ID` attributes serve the same function in their respective data layers, but they are not related. A value of `1` for the `ID` attribute in `Layer 1` has nothing to do with the `ID` value of `1` in `Layer 2`. It simply identifies a unique combination of attributes in the `output layer`.

Vector overlay of these two polygon data layers results in four new polygons. Each new polygon contains the attribute information from the corresponding area in the input data layers. For example, note that the polygon in the output data layer with the `ID` of `1` has a `class` attribute with a value of `0` and a `cost` attribute with a value of `10`. These values come from the values found in the corresponding input layers. The boundary for the polygon with an `ID` value of `1` in the output data layer is a composite of the boundaries found in the two input data layers. The same holds true for the other three polygons in the output data layer. These polygons are a composite of geographic and attribute data in the input data layers.

The topology of vector overlay output will likely be different from that of the input data layers. Vector overlay functions typically identify line intersections during

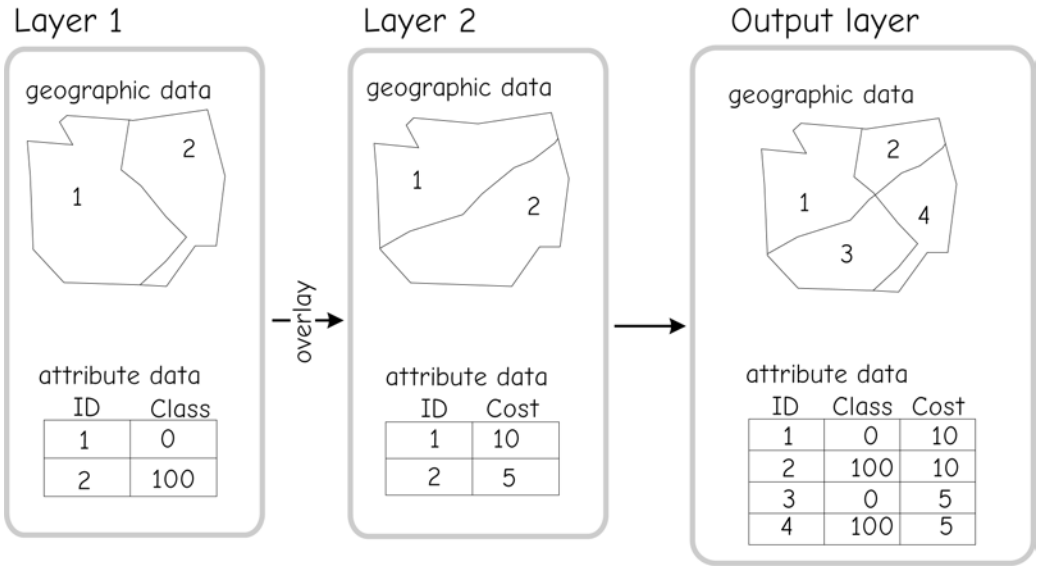


Figure 9-34: An example of vector polygon overlay. In this example, output data contain a combination of the geographic (coordinate) data and the attribute data of the input data layers. New features may be created with topological relationships distinct from those found in the input data layers.

overlay. Intersecting lines are split and a node placed at the intersection point. Thus topology must be recreated if it is needed in further processing.

Any type of vector feature may be overlaid with any other type of vector feature, although some overlay operations rarely provide useful information and are performed infrequently. In theory, points may be overlaid on point, line, or polygon feature layers, lines on all three types, and polygons on all three types. Point-on-point or point-on-line overlay rarely results in intersecting features, and so they are rarely applied. Line-on-line overlay is sometimes required, for example, when we wish to identify the intersections of two networks such as road and railroads, but these also are rare occurrences. Overlays involving polygons are the most common by far.

Overlay output typically takes the lowest dimension of the inputs. This means point-in-polygon overlay results in point output, and line-in-polygon overlay results in line output. This avoids problems when multiple lower dimension features intersect with higher dimension features.

Figure 9-35 illustrates an instance where multiple points in one layer fall within a single polygon in an overlay layer. Output attribute data for a feature are a combination of the input data attributes. If polygons are output (Figure 9-35, right,

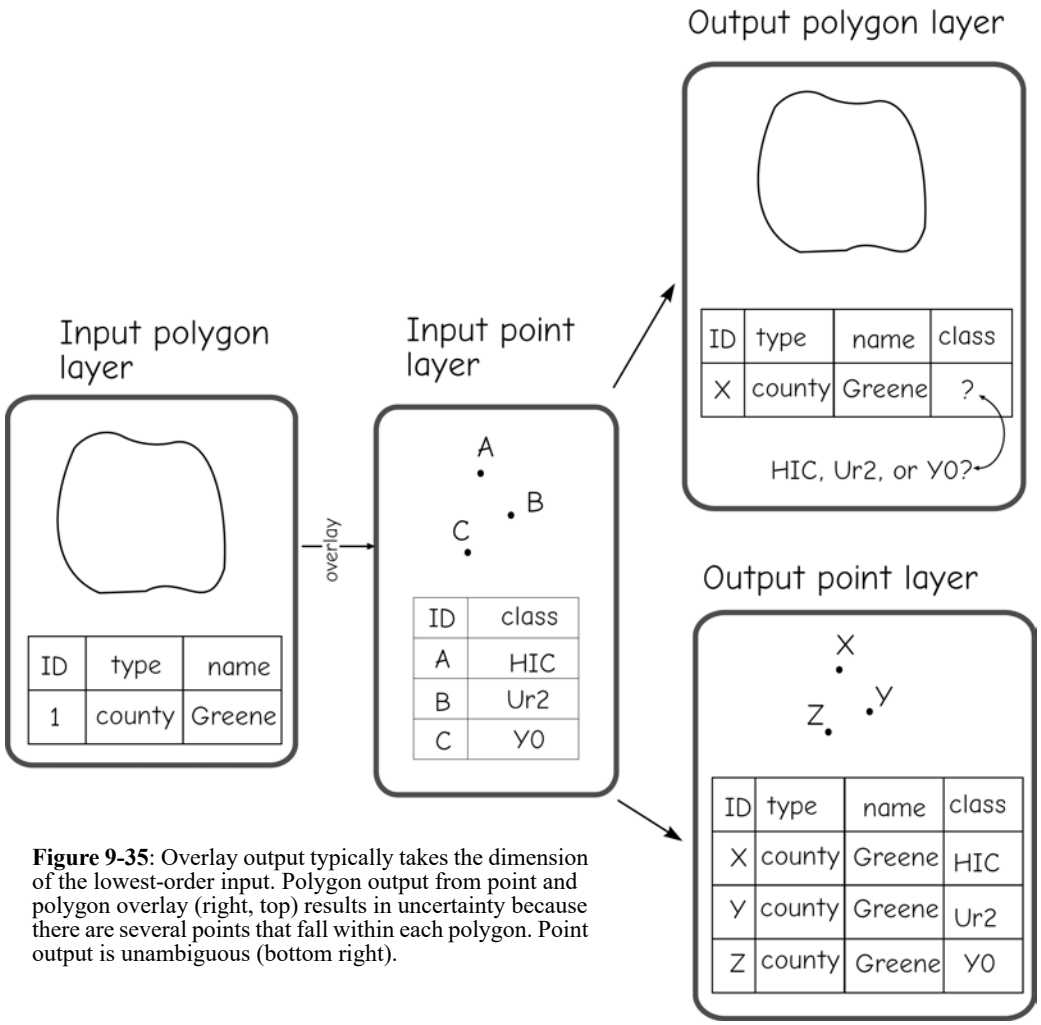


Figure 9-35: Overlay output typically takes the dimension of the lowest-order input. Polygon output from point and polygon overlay (right, top) results in uncertainty because there are several points that fall within each polygon. Point output is unambiguous (bottom right).

top), there is ambiguity regarding which point attribute data to record. Each point feature has a value for an attribute named class. It is not clear which value should be recorded in the output polygon, the class value from point A, point B, or point C. When a point layer is output (Figure 9-35, right, bottom), there is no ambiguity. Each output point feature contains the original point attribute information, plus the input polygon feature attributes.

One method for creating polygon output from point-in-polygon overlay involves recording the attributes for one point selected arbitrarily from the points that fall within a polygon. This is usually not satisfactory because important information may be lost. An alternative involves adding columns to the output polygon to preserve multiple points per polygon. However, this would still result in some ambiguity, such as, what should be the order of duplicate attributes? It may also add a substantial number of sparsely used items, thus increasing file size inefficiently. Forcing the lower order output during overlay avoids these problems, as shown in the lower right of Figure 9-35.

Note that the number of attributes in the output layer increases after each overlay. This is illustrated in Figure 9-35, with the combination of a point and polygon layer in an overlay. The output point attribute table shown in the lower right portion of the figure contains four items. This output attribute table is a composite of the input attribute tables.

Large attribute tables may result if overlay operations are used to combine many data layers. When the output from an overlay process is in turn used as an input for a subsequent overlay, the number of attributes in the next output layer will usually increase. As the number of attributes grows, tables may become unwieldy, and we often delete redundant attributes.

Overlays that include a polygon layer are most common. We are often interested in the combination of polygon features with other polygons, or in finding the coincidence of point or line features with polygons. What counties include hazardous waste sites? Which neighborhoods does one pass through on E Street? Where are there shallow aquifers below cornfields? All these examples involve the overlay of area features, either with other area features, or with point or line features.

Clip, Intersect, and Union: Special Cases of Overlay

There are three common ways overlay operations are applied: as a *clip*, an *intersection*, or a *union*.

The basic layer-on-layer combination is the same for all three. They differ in the geographic extent for which vector data are recorded, and in how data from the attribute layers are combined. Intersection and union are derived from general set theory operations. The intersection operation may be considered in some ways to be a spatial AND, while the union operation is related to a spatial OR. The clip operation may be considered a combination of an intersection and an elimination. All three are common and supported in some manner as stand-alone functions by most GIS software packages.

A *clip* may be considered a "cookie-cutter" overlay (Figure 9-36). A bounding polygon layer is used to define the areas for which features will be output. This bounding polygon layer defines the clipping region. Point, line, or polygon data in a second layer are "clipped" with the bounding layer. In most versions of the clip function, the attributes for the clipping layer are not included in the output data layer. A clip is most often used when sub-setting data geographically, to reduce data volumes when working on a small area included in larger data layers. A city manager may only wish the set of streets within their city boundaries, clipped from a statewide roads layer.

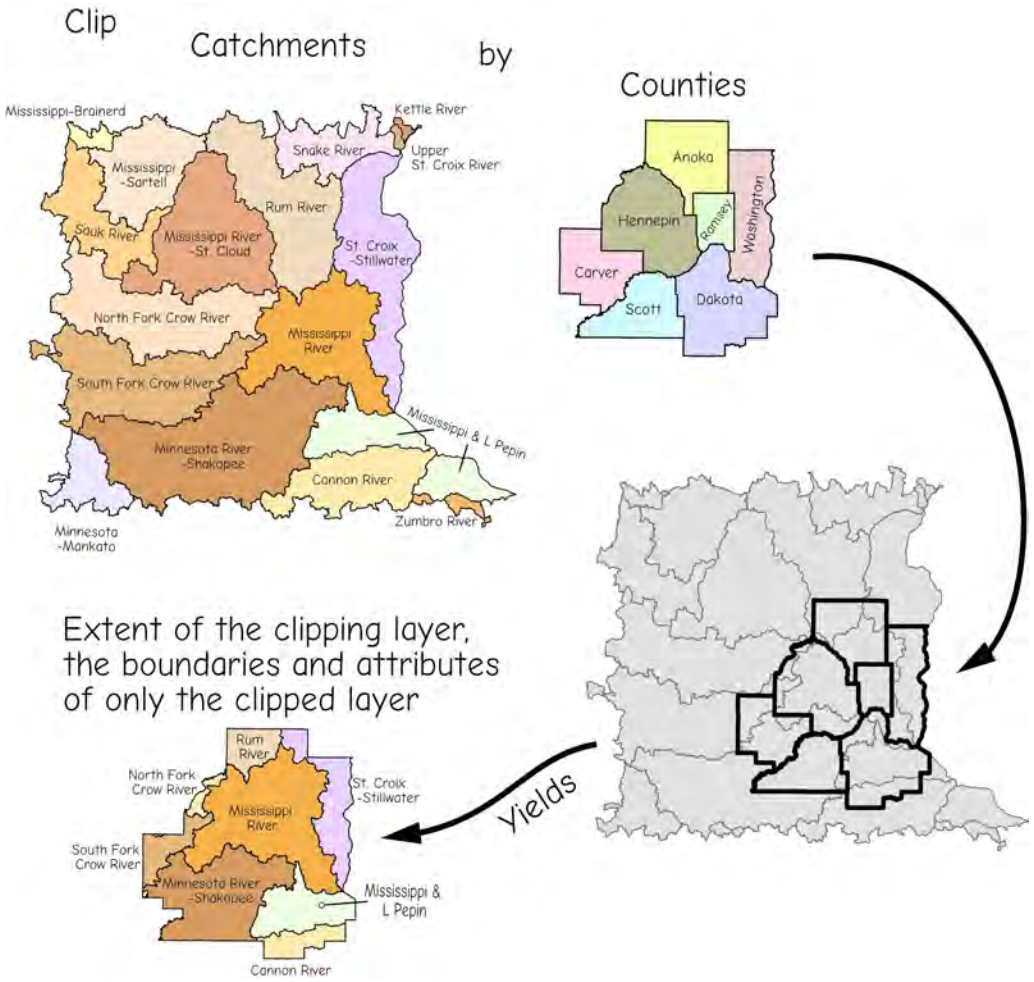


Figure 9-36: A clip is a common variation of overlay operations. The clip preserves information only from the clipped (or target) data layer and only for the area of the clipping (or bounding layer). The attribute table of the output layer typically contains all the attributes of the clipped layer (here, Catchments), and none from the clipping layer (Counties).

In our example shown in Figure 9-36, the bounding or clipping data layer consists of seven county polygons, and the target or clipped data layer contains many small catchment boundaries. The presence of polygon attributes in the bounding layer is indicated by the different shades for the different county polygons. The output from the clip consists of those portions of catchments within the clip layer boundary. Note that the clip layer boundaries, here counties, are not included in the output data layer. Also note that only the attributes for the clipped catchment layer are output.

Users should be certain that transferred variables still have valid values after a clip. If an area field is included in the input layer, the value may be wrong if it is not re-calculated. Other area-based values may also be in error, e.g., a polygon density or counts of included features. Since software defaults vary, the behavior of the specific software tool should be identified.

An *intersection* is another multi-layer combination, and may be defined as an overlay that fuses data from both layers, but only for the area where both layers contain data (Figure 9-37). Features boundaries from both data layers are combined. Both layers serve as data and as bounding layers, so that any parts of polygons that are in one layer but not another are clipped and discarded.

The same caution on relevancy and recalculation of combined, area-based variables applies to the output from intersection operations as applied to clip operations. Since new geographies with differing areas are created, attributes transferred from the component features may need new interpretations. Most implementations simply copy the values from the input features to the coincident output features. For example, a county area for each polygon will come from the input county polygon. While this was valid for each input county in Figure 9-37, it will be replicated for each polygon in the output layer, and should be interpreted as the area for the contributing county, not the area from that county in the (usually smaller) output polygon. Summing across polygons with a

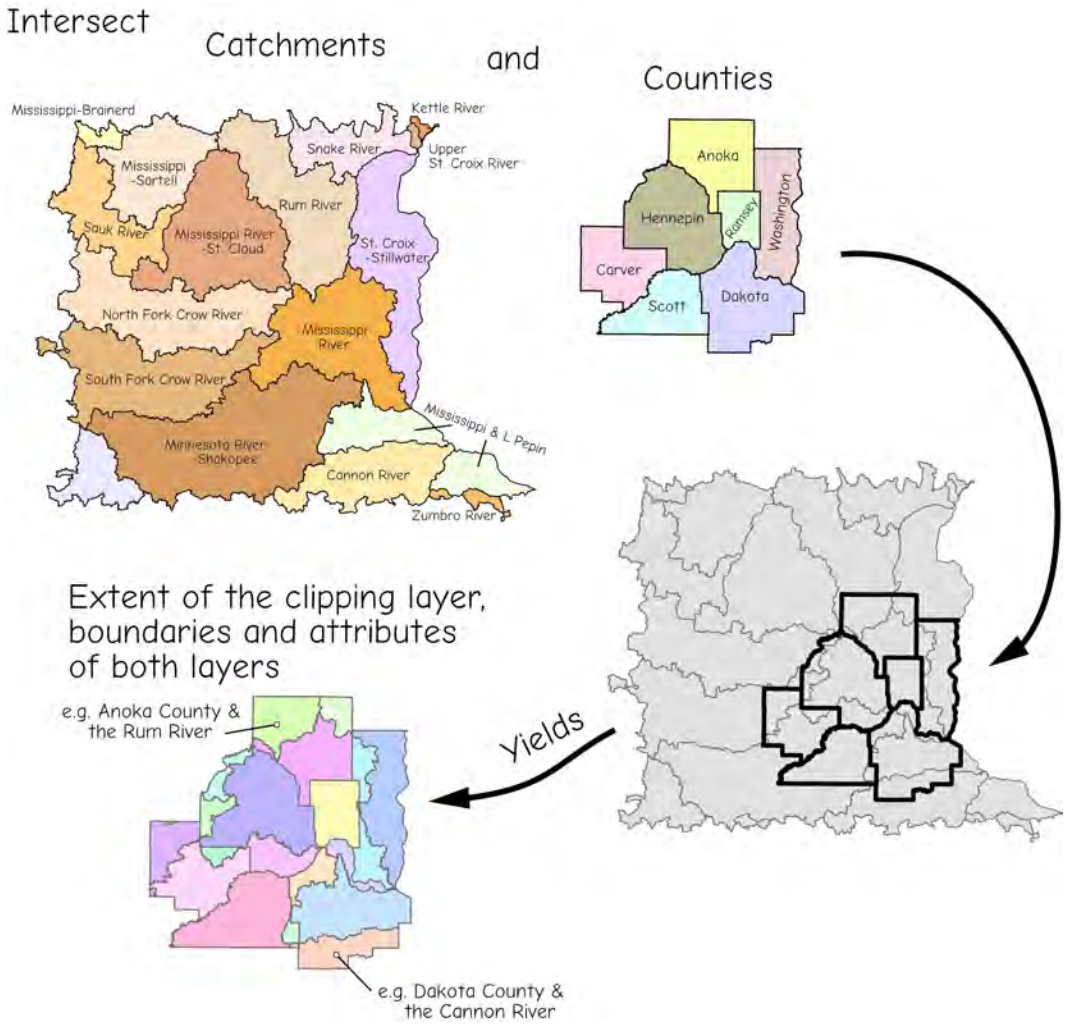


Figure 9-37: An intersection is another common overlay operation. Both the boundaries and data are combined in the output, but only for the areas that are contained in the clipping layer, Counties in this example. The particular software tool typically requires you explicitly identify the clipping layer.

given county value will usually not give an accurate county area. Each output variable should be scrutinized and the value origin and contents clearly understood.

A *union* is another kind of overlay. It retains all data from both the bounding and data layers (Figure 9-38). No geographic data are discarded in the union operation, and corresponding attribute data are saved for all regions. New polygons are formed by the combination of coordinate data from each data layer.

Note that there are often many null or empty attribute values in union output layers. Data in non-overlapping areas are absent and so cannot be assigned, e.g., outside the county layer bounds but within the watershed layer bounds in Figure 9-38. The presence of null values may alter subsequent operations.

Many software packages support additional variants of overlay operations. Some support an *Erase* or similarly-named function, which is the complement to the clip function. In an Erase function, areas covered by the input layer are "cut out" or

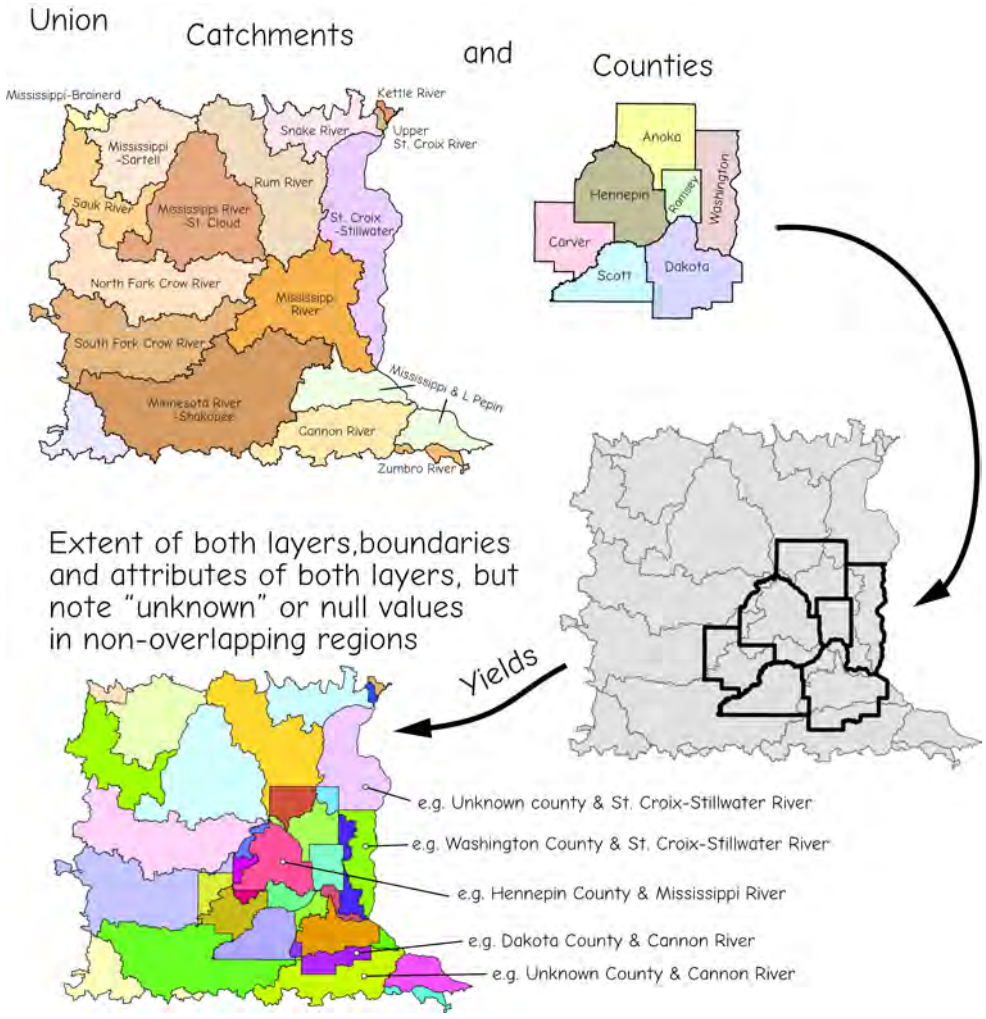


Figure 9-38: A union operation. Data from two layers are combined, including any areas that are contained by one but not both data layers. No areas nor attribute data are discarded, or "clipped," as in the intersection and clip operations. Blank or null values are typically assigned in the areas contained in one but not the other layer, e.g., in the example where the county is listed as "Unknown."

erased from the bounding layer (Figure 9-39). Erasures may cut existing polygons apart or, where there are coincident lines in the two data layers, may preserve the edge of existing polygons. In some versions, there is a tolerance distance that allows for lines that aren't exactly coincident in different layers, but are meant to be, to be represented only once in the output. This tolerance distance effectively serves as a snapping distance, and moves the vertices in one layer on a nearly coincident edge to match vertices in the other layer. As with snapping during digitization or other overlays, this may help reduce incorrect or unwanted geometries.

The *Erase* function is particularly useful when updating a portion of a data layer, in that old, out of date, poorer quality, or otherwise inferior data may be clipped out of a section and newer data substituted. Erase is also often useful in spatial analyses that include criteria specifying areas that are greater than some distance from a set of

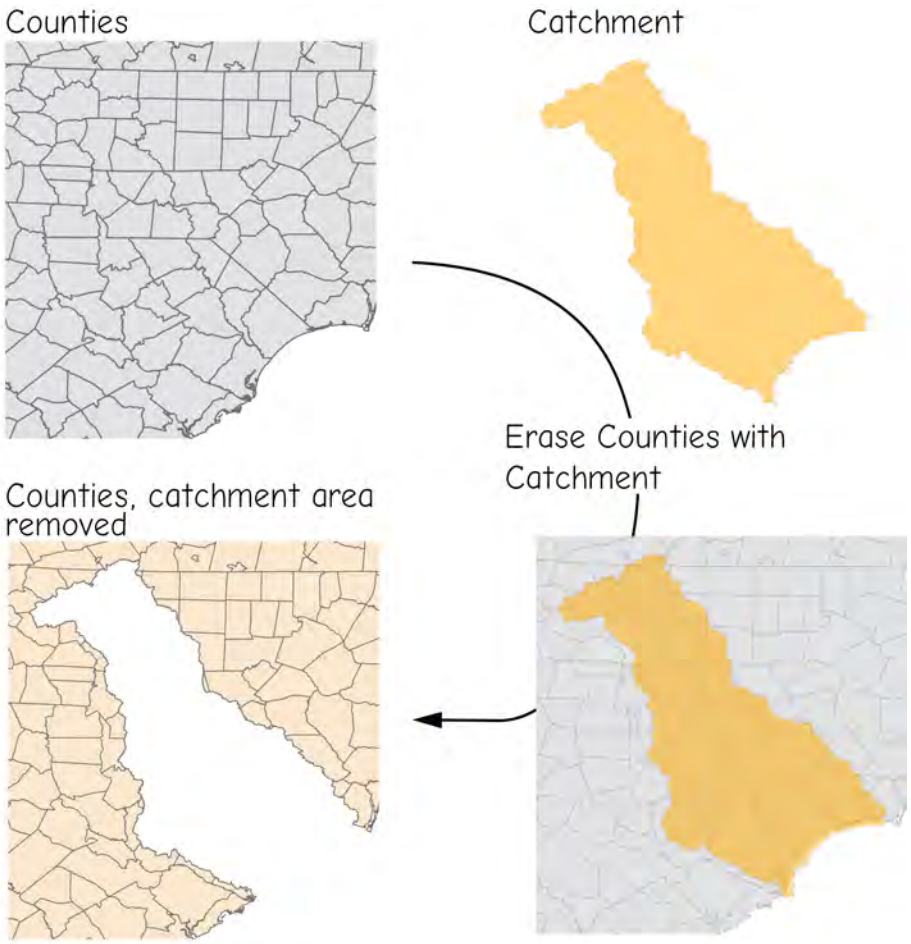


Figure 9-39: An example of an Erase operation. Features in an input layer are "erased" based on the outer boundary of an erasing polygon. This operation is often used in editing or cartographic models that specify area removal based on a buffered layer.

features. A buffer function identifies areas that are less than the target distance, and these may then be removed from consideration using an erase operation.

There are other variants on unions or intersections. Most of these specialized overlay operations may be created from the application of union or overlay operations in combination with selection operations.

Vector overlay is often a time-consuming computational process, due to the large number of lines that must be compared. A vector overlay typically requires repeated tests of line intersection, a relatively simple set of calculations, but there is often a large number of line segments in a data set. Each line segment must be checked against every other line segment, requiring perhaps billions of tests for line intersection.

A Problem in Vector Overlay

Polygon overlays often suffer when there are common features that are represented in both input data layers. We define a common feature as a different representation of the same phenomenon. Figure 9-40 illustrates this problem. A county boundary may coincide with a state boundary. However, different versions of the state and county boundaries may be created independently from two adjacent states, using different source materials, at different times, and using different systems. Thus, these two representations may differ even though they identify the same boundary on the earth surface.

In most data layers, the differences will be quite small, and will not be visible except at very large display scales, for example, when the on-screen zoom is quite

Figure 9-40: Sliver polygons may occur when two representations of a feature are combined. A common boundary between two features has been derived from different sources. The representations differ slightly. This results in small gaps and "sliver" polygons along the margin between these two layers.

high. The differences are shown in the larger-scale inset in Figure 9-40. When the county and state data layers are overlaid, many small polygons are formed along the boundary. These polygons are quite small, but they are often quite numerous.

These "sliver" polygons cause problems because there is an entry in the attribute table for each polygon. One-half or more of the polygons in the output data layer may be these slivers. Slivers are a burden because they take up space in the attribute table but are not of any interest or use. Analyses of large data sets are hindered because all selections, sorts, or other operations must treat all polygons, including the slivers. Processing times often increase exponentially with the number of polygons.

There are several methods to reduce the occurrence of these slivers. One identifies all common boundaries across different layers. The boundary with the highest coordinate accuracy is substituted into all other data layers, replacing the less accurate representations. This involves considerable editing, and most common when developing new data layers.

Another method involves manually identifying and removing slivers. Small polygons may be selected, or polygons with two bounding arcs, common for sliver polygons. Bounding lines may then be adjusted or removed. However, manual removal is not practical for many data sets due to the high number of sliver polygons.

A third method for sliver reduction involves defining a snap distance during overlay. Much as with a snap distance used during data development (described in Chapter 4), this forces nodes or lines to be coincident if they are within a specified proximity during overlay. As with data entry, this snap distance should be small relative to the spatial accuracy of the input layers and the required accuracy of the output data layers. If the two representations of a line are within the snap distance then there will be no sliver polygons. In practice, not all sliver polygons are removed, but their numbers are substantially reduced, thereby reducing the time spent on manual editing.

Automatic sliver detection and removal should be applied carefully, as they may delete valuable data. Only small slivers should be removed, with small defined as smaller than an area, length, or width that is worth tracking. This distance may be set by the accuracy of the data collection, or by the requirements of the analysis. If polygon edge locations are only digitized to within one meter of their true position, it makes little sense to maintain polygons that are less than a meter in any dimension. However, if slivers are removed that are substantially wider and longer than a meter, some valuable information may be lost.