



## **FRAGSTATS SPATIAL PATTERN ANALYSIS PROGRAM**

**DOCUMENT WAS WRITTEN BY**

**DR KEVIN MCGARIGAL**

**AND CAN BE FOUND AT**

<http://www.umass.edu/landeco/research/fragstats/fragstats.html>

## **What is FRAGSTATS?**

FRAGSTATS is a spatial pattern analysis program for categorical maps. The landscape subject to analysis is user-defined and can represent any spatial phenomenon. FRAGSTATS simply quantifies the areal extent and spatial configuration of patches within a landscape; it is incumbent upon the user to establish a sound basis for defining and scaling the landscape (including the extent and grain of the landscape) and the scheme upon which patches are classified and delineated. We strongly recommend that you read the FRAGSTATS Background document on landscape pattern metrics (also available at this web site). The output from FRAGSTATS is meaningful only if the landscape mosaic is meaningful relative to the phenomenon under consideration.

## **Scale Considerations**

FRAGSTATS does not limit the scale (extent or grain) of the landscape subject to analysis. However, the distance- and area-based metrics computed in FRAGSTATS are reported in meters and hectares, respectively. Thus, landscapes of extreme extent and/or resolution may result in rather cumbersome numbers and/or be subject to rounding errors. However, FRAGSTATS outputs data files in ASCII format that can be manipulated using any database management program to rescale metrics or to convert them to other units (e.g., converting hectares to acres).

## **Computer Requirements**

FRAGSTATS version 3 is a stand-alone program written in Microsoft Visual C++ for use in the Windows operating environment. FRAGSTATS was developed and tested on the Windows NT and 2000 operating systems, although it should run under all Windows operating systems. Note, FRAGSTATS is highly platform dependent, as it was developed in the Microsoft environment, so portability to other platforms is not easily accomplished. FRAGSTATS is a compute-intensive program; its performance is dependent on both processor speed and computer memory (RAM). Ultimately, the ability to process an image is dependent on the availability of sufficient memory, the speed of processing that image is dependent on processor speed. Of particular note is the memory constraint. FRAGSTATS loads the input grid into memory and then computes all requested calculations. Thus, you must have sufficient memory to load the grid and then enough leftover for processing and other operating system needs. To determine whether you have sufficient memory to load a particular grid, you can use the following formula: #cells\*4bytes. Thus, if you have a 256 rows by 256 columns grid, the memory requirement to simply load the grid is 256 kb ( $256*256*4/1024$  bytes/kb). Plus you need sufficient memory to process the grid, and this requirement depends on the number of patches and classes. These memory requirements are not particularly constraining in a standard analysis, unless you are working with huge images on an older computer. The solution to this problem if it arises—unfortunately—is to get a new machine with as much memory as possible. The memory requirement, however, is very constraining in the moving window analysis because FRAGSTATS requires enough memory for the input grid plus one output grid, plus enough leftover for other processing needs and system needs. If the moving window analysis is selected, FRAGSTATS checks to see if it can allocate enough memory for three grids (i.e., 1 input grid + 1 output grid + enough leftover to insure performance). In the above example, you would need at least 768 kb of

memory to conduct a moving window analysis. Not too terribly constraining for most computers when analyzing relatively small landscapes. However, consider a 10,000x10,000 input grid; to conduct a moving window analysis you would need at least 1.14 Gb of RAM.

## Data Formats

FRAGSTATS accepts raster images in a variety of formats, including ArcGrid, ASCII, BINARY, ERDAS, and IDRISI image files. FRAGSTATS does not accept Arc/Info vector coverages like the earlier version 2. All input data formats have the following common requirements:

- All input grids should be *signed integer* grids containing all **non-zero** class values (i.e., each cell should be assigned an integer value corresponding to its class membership or patch type). Note, FRAGSTATS assumes that the grids are ‘signed’ integer grids; inputting an ‘unsigned’ integer grid may cause problems. In addition, note that assigning the zero value to a class is allowable, but can cause problems in the moving window analysis if it is not specified as the background class. Specifically, if zero is specified as the background class, FRAGSTATS will reclassify all zero cell values to a new class value equal to 1 plus the largest class value in the input landscape. This procedure is necessary because a zero background class value will cause problems in the moving window analysis because a background value of zero in the output grid cannot be distinguished from a computed metric value of zero. In addition, a zero class value will cause problems when the landscape contains a border because zero cannot be negative, yet all cells in the border must be negative integers). Thus, all zero cells are assumed to be interior (i.e., inside the landscape of interest). If a moving window analysis is selected and the input grid contains zeros and zero is NOT specified as the background class, then execution will be stopped and an error message reported to the log file. For these reasons, it is best to avoid the use of zero class values altogether.
- All input grids should consist of *square* cells with cell size specified in *meters*. For certain input formats (ASCII and BINARY), this is not an issue because cells are assumed to be square and you are required to enter the cell size (in meters) in the graphical user interface. However, FRAGSTATS assumes that all other image formats (ArcGrid, ERDAS, and IDRISI) include header information that defines cell size. Consequently these images must have a metric projection (e.g., UTM) to ensure that cell size is given in metric units.

There are some additional special considerations for each input data format, as follows:

(1) **Arc Grid** created with Arc/Info. Note, to use Arc Grids you must have ArcView Spatial Analyst or ArcGIS installed on your computer and FRAGSTATS must have access to a certain .dll file found either in the ArcView Bin32 directory (for ArcView Spatial Analyst users) or the ArcGIS Bin directory (for ArcGIS users). Specifically, a path to the corresponding dll library file should be specified in the environmental settings

under NT or Windows 2000 operating systems, or a path statement included in the autoexec.bat file, e.g., under Windows 98, as follows:

Windows NT: You can add the necessary Path variable or edit the existing one via the Control panel - System Properties - Environment tab. Add a new variable or edit the existing Path variable in the system variables, not the user variables (this will require administrative privileges). Add the full path to the appropriate .dll file. If you are using ArcView Spatial Analyst, the required file is the avgridio.dll file and it is typically installed in the following path: \esri\av\_gis30\arcview\bin32. If you are using ArcGIS, the required file is the agridio.dll file and it is typically installed in the following path: \esri\arcinfo\arcexe81\bin. Note, your software version number and path may be different so be sure to locate the .dll file on your computer and enter the correct path. If you are using both Spatial Analyst and ArcGIS, then you can enter either or both paths to the Path system variable.

Windows 2000/XP: You can add the necessary Path variable or edit the existing one via the Control panel - System Properties - Advanced tab - Environment Variables button. Add a new variable or edit the existing Path variable in the system variables, not the user variables (this will require administrative privileges). Then, following the instructions above for Windows NT.

Windows 95/98: You must add the necessary Path statement to the autoexec.bat file. First, search your computer for the autoexec.bat file and open it using any text editor. Then, either add a Path statement or edit the existing one. Add the full path to the appropriate .dll file. If you are using ArcView Spatial Analyst, the required file is the avgridio.dll file and it is typically installed in the following path: \esri\av\_gis30\arcview\bin32. If you are using ArcGIS, the required file is the agridio.dll file and it is typically installed in the following path: \esri\arcinfo\arcexe81\bin. Thus, the path statement should look something like: PATH c:\esri\av\_gis30\arcview\bin32. Note, your software version number and path may be different so be sure to locate the .dll file on your computer and enter the correct path. If you are using both Spatial Analyst and ArcGIS, then you can enter either or both paths to the Path system variable. If you are adding the path to an existing path, simple use a semicolon to separate the unique paths in the Path statement. After saving the file you will need to reboot your machine for the change to take effect.

(2) **ASCII** file, no header. Each record should contain 1 image row. Cell values should be separated by a comma or a space(s). Note, it will be necessary to strip (delete) the header information from the image file if it exists, but be sure to keep it for later reference regarding background cell value, # rows, and # columns.

(3) **32-bit binary** file, no header; no other limitations.

(4) **16-bit binary** file, no header. Patch ID output file, if selected, will be output in *signed 32-bit integer* format to accommodate a greater number of patches. In addition, because

moving window analysis requires floating points, if moving window analysis is selected, the output grids produced will be *32-bit floating point* grids.

(5) **8-bit binary** file, no header. Patch ID output file, if selected, will be output in *signed 32-bit integer* format to accommodate a greater number of patches. In addition, because moving window analysis requires floating points, if moving window analysis is selected, the output grids produced will be *32-bit floating point* grids.

(5) **ERDAS** image files (.gis, .lan, and .img). FRAGSTATS accepts images from both ERDAS 7 (.gis and .lan) and ERDAS 8 (.gis, .lan, and .img), but the limitations are somewhat different, as follows:

ERDAS 8 Files.—FRAGSTATS accepts .gis, .lan, and .img files used by current versions of ERDAS IMAGINE, including *signed 8-, 16-, and 32-bit integer* grids. While .gis and .lan file formats are supported, their limitations make them less practical than .img (see discussion of ERDAS 7 files below). Care should be taken when preparing the data to be used with FRAGSTATS, especially when importing data from other formats (for instance from Arc Grid). Be sure to set the import options to '*signed integer*'. Regardless of the input integer format (8-, 16-, or 32-bit), the patch ID output file created by FRAGSTATS will be a *signed 32-bit integer*, and if moving window analysis is selected, all output grids will be *32-bit floating point*. Multi-layered files are not rejected, but only layer one is processed and the outputs are all single layered. As noted above, cells must be square, not rectangular, and the measurement unit should be meters. If the measurement unit is not specified, then it is assumed to be meters. The cell size and measurement unit specification can be changed within ERDAS Imagine using the ImageInfo tool -> Change Map Model from the Edit menu. The projection information existing in the input file is passed unchanged to the output files. FRAGSTATS does not use this information internally.

ERDAS 7 files.-- FRAGSTATS accepts .gis and .lan files used by ERDAS 7, which are limited to *unsigned 8- or 16-bit integer* grids. FRAGSTATS will accept both *8-bit* and *16-bit integer* files and it will reject multi-layered files. While .gis and .lan file formats are supported, their limitations make them less practical than .img. In particular, ERDAS 7.x files are limited to *unsigned integers* (i.e., only positive integers), therefore landscape borders (which require negative class values) cannot be represented. Another consequence of this particular limitation is that FRAGSTATS-generated patch ID files will not fill the non-landscape cells (i.e., no data cells) with the usual value (minus the background class value), but with zero values. The restriction to 8- or 16-bit integers imposes some limitations when using the FRAGSTATS on large grids. Specifically, unsigned 16-bit integers can only take on values up to 65,534. Thus, class ID values are limited to integers within this range (note, this should not be a problem, since it is unlikely that anyone would have more than 65,534 patch types). Similarly, patch ID values in the patch ID output grid optionally produced by FRAGSTATS are limited to the same range, effectively limiting the number of patches in this output grid to 65,534. If the grid contains more than this number of patches, FRAGSTATS will not be able to output a unique ID for

each patch and the user will have to somehow distinguish among patches with the same ID. Finally, because the moving window analysis requires 32-bit output files (in order to output floating point values), moving window analysis is not supported with ERDAS 7 files.

(6) **IDRISI** image files (.rdc). IDRISI currently supports *signed 8- or 16-bit integers* and *32-bit floating point* grids. This imposes some limitations when using the FRAGSTATS on large grids. Specifically, signed 16-bit integers can only take on values between -32,768 and 32,767. Thus, class ID values are limited to integers within this range (note, this should not be a problem, since it is unlikely that anyone would have more than 32,767 patch types). Similarly, patch ID values in the patch ID output grid optionally produced by FRAGSTATS are limited to the same range, effectively limiting the number of patches in this output grid to 37,767. If the grid contains more than this number of patches, FRAGSTATS will not be able to output a unique ID for each patch and the user will have to somehow distinguish among patches with the same ID. Fortunately, IDRISI supports 32-bit floating point grids, which are produced in a moving window analysis.

## **Raster Considerations**

It is important to realize that the depiction of edges in raster images is fundamentally constrained by the lattice grid structure. Consequently, raster images portray lines in stair-step fashion. The result is an upward bias in the measurement of edge length; that is, the measured edge length is always more than the true edge length. The magnitude of this bias depends on the grain or resolution of the image (i.e., cell size), and the consequences of this bias with regards to the use and interpretation of edge-based metrics must be weighed relative to the phenomenon under investigation.

## **Vector to Raster Conversion**

In some investigations, it may be necessary to convert a vector coverage into a raster image in order to run FRAGSTATS. It is critical that great care be taken during the rasterization process and that the resulting raster image be carefully scrutinized for accurate representation of the original image. During the rasterization process, it is possible for disjunct patches to join and vice versa. This problem can be quite severe (e.g., resulting in numerous 1-cell patches) if the cell size chosen for the rasterization is too large relative to the minimum patch dimension in the vector image. In general, a cell size less than  $\frac{1}{2}$  the narrowest dimension of the smallest patches is necessary to avoid these problems.

## **Levels of Metrics**

FRAGSTATS computes 3 groups of metrics. For a given landscape mosaic, FRAGSTATS computes several metrics for: (1) each patch in the mosaic; (2) each patch type (class) in the mosaic; and (3) the landscape mosaic as a whole. These metrics are described in detail in the FRAGSTATS Metrics documentation. In addition, FRAGSTATS computes the adjacency matrix (i.e., tally of the number of cell adjacencies between each pairwise combination of patch types, including like-adjacencies between cells of the same class), which is used in the computation of several class- and landscape-level metrics.

## Output Files

Depending on which metrics are selected by the user, FRAGSTATS creates 4 output files corresponding to the three levels of metrics and the adjacency matrix. The user supplies a "basename" for the output files and FRAGSTATS appends the extensions .patch, .class, .land and .adj to the basename. All files created are *comma-delimited ASCII files* and viewable. These files are formatted to facilitate input into spreadsheets and database management programs:

- **"basename".patch file.**—Contains the patch metrics; the file contains 1 record (row) for each patch in the landscape; columns represent the selected patch metrics. If a batch file is analyzed, the file contains 1 record for each patch in each landscape specified in the batch file. The first record is a column header consisting of the acronyms for all the metrics that follow. For a single landscape, the patch output file would be structured as follows:

LID,	PID,	TYPE,	AREA,	PERIM,	GYRATE,	CORE
D:\testgrid,	9,	forest,	1.0000,	400.0000,	38.1195,	0.1600
D:\testgrid,	0,	shrub,	4.0000,	800.0000,	76.4478,	1.9600
Etc.						

- **"basename".class file.**—Contains the class metrics; the file contains 1 record (row) for each class in the landscape; columns represent the selected class metrics. If a batch file is analyzed, the file contains 1 record for each class in each landscape specified in the batch file. The first record is a column header consisting of the acronyms for all the metrics that follow. For a single landscape, the class output file would be structured as follows:

LID,	TYPE,	CA,	PLAND,	NP,	PD,	LPI
D:\testgrid,	forest,	8.0000,	22.5000,	4,	5.0000,	15.000
D:\testgrid,	shrub,	21.0000,	26.2500,	3,	3.7500,	12.500
Etc.						

- **"basename".land file.**—Contains the landscape metrics; the file contains 1 record (row) for the landscape; columns represent the selected landscape metrics. If a batch file is analyzed, the file contains 1 record for each landscape specified in the batch file. The first record is a column header consisting of the acronyms for all the metrics that follow. For a single landscape, the landscape output file would be structured as follows:

LID,	TA,	NP,	PD,	LPI,	TE,	ED
D:\testgrid,	80.0000,	12,	15.0000,	15.0000,	7800.0000,	97.500

- **"basename".adj file.**—Contains the class adjacency matrix; the file contains a simple header in addition to 1 record (row) for each class in the landscape, and is given in the form of a 2-way matrix. Specifically, first record contains the input file name, including the full path. The second record and first column contain the class IDs (i.e., the grid integer values associated with each class), and the elements of the matrix are the tallies of cell adjacencies for each pairwise combination of classes. For a single landscape, the adjacency output file would be structured as follows:

D:\testgrid					
Class ID / ID,	2,	3,	4,	5,	Background
2,	6840,	130,	120,	10,	0

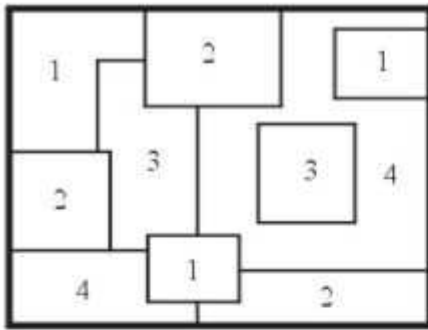
3,	120,	7960,	160,	40,	10
4,	100,	140,	9880,	40,	20
5,	10,	40,	30,	3080,	16

Note, the adjacency tallies are generated from the *double-count method* in which each cell side is counted twice—at least for all positively-valued nonbackground cells—and only the *4 orthogonal neighbors* are considered. In addition, the matrix may not be symmetrical if a landscape border is present because landscape boundary edges are only counted once. For example, a cell of class 3 (inside the landscape) adjacent to a cell of class -5 (in the landscape border) results in an adjacency for class 3; specifically, a 3 (row) -5 (column) adjacency. It does not result in an adjacency for class 5 (row), because the border cells themselves are not evaluated. For this reason, the adjacency matrix must be read as follows: each row represents the adjacency tallies for cells of that class, and the sum of adjacencies across all columns represents the total number of adjacencies for that class. These *row totals* should equal the number of positively-valued cells (i.e., inside the landscape) of the corresponding class times 4 (i.e., 4 surfaces for each cell). These row totals are used in several of the contagion metrics. Note that the adjacency matrix includes a column for background adjacencies, which represent cell surfaces of the corresponding class adjacent to designated background. If there is no specified background in the input landscape and a landscape border is not present, then the background adjacencies represent the cell surfaces along the landscape boundary—which are treated as background in the absence of a border. If a border is provided and no background is specified, then the background adjacencies will equal zero because every cell surface, including those along the landscape boundary, will be adjacent to a real non-background class. If a batch file is analyzed, the adjacency matrices corresponding to each landscape specified in the batch file are appended to the same file.

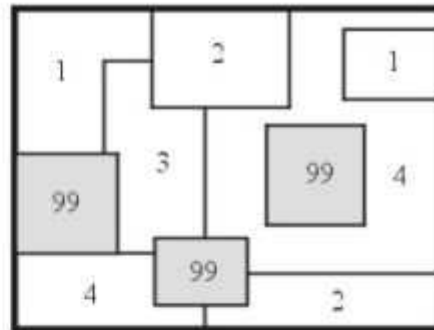
### **Backgrounds, Borders, and Boundaries**

FRAGSTATS accepts images in several forms, depending on whether the image contains **background**, and whether the landscape contains a **border** outside the landscape **boundary** (Fig. 1). The distinction among background, border, and boundary and how they affect the landscape analysis and the calculations of various metrics is a source of great confusion. Care should be taken to fully understand these distinctions before trying to run FRAGSTATS. We strongly suggest that you read through this section twice, once to familiarize yourself with the terminology and a second time to understand the implications.

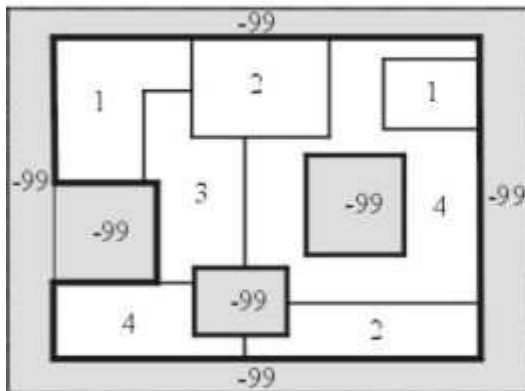
A. No background/no border



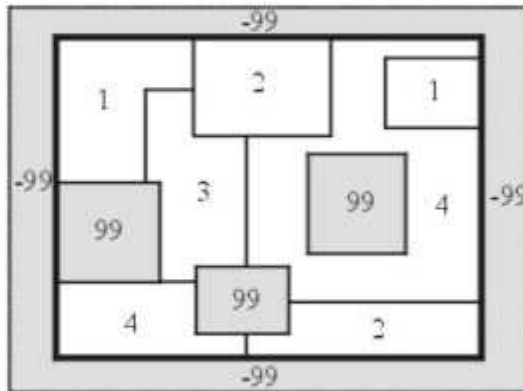
B. Interior background/no border



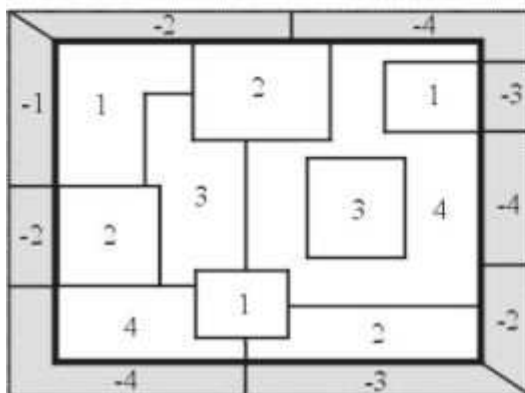
C. Exterior background/no border



D. Interior/Exterior background/no border



E. No background/with border



F. Interior/exterior background/with border

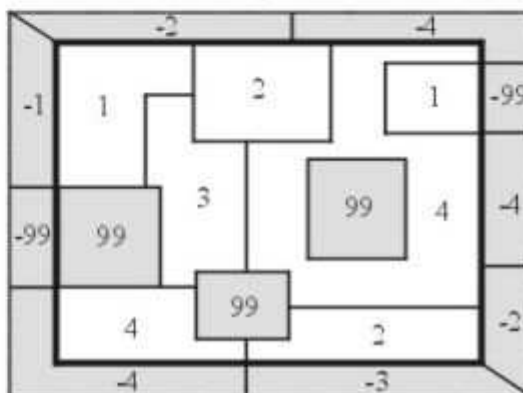


Figure 1. Alternative image formats with regards to background (given a class value of 99 here) and border. The thick solid line represents the landscape boundary. Positive values are 'inside' the landscape of interest and contribute to the computed total landscape area; negative values are 'outside' the landscape of interest and are only utilized to determine edge types for patches along the landscape boundary.

Every image will include a **landscape boundary** that defines the perimeter of the landscape and surrounds the patch mosaic of interest. The boundary is simply an invisible line around the patch mosaic of interest. It is not given explicitly in the image; rather, it is defined by an imaginary line around the outermost cells of positively valued cells. The

landscape boundary distinguishes between cells *inside* the landscape of interest from those *outside*, and thus ultimately determines the total landscape area. All positively valued cells are assumed to be *inside* the landscape of interest and are thus included in the total area of the landscape, regardless of whether they are classified as background (see below) or not. This is important, because many metrics include total landscape area in their calculation. Note, in most cases the landscape boundary will surround a single contiguous region of positively valued cells. However, it is possible to have disjunct regions of positively valued cells. In this case, the landscape boundary is not a single continuous line around the landscape of interest, but rather separate boundaries around each disjunct region of interest. The important point is that positively valued cells are *inside* the landscape of interest, while negatively valued cells are *outside*, and the landscape boundary is the imaginary line(s) that separate *inside* from *outside*. Hence, if the input image contains all positively valued cells, then the entire grid is assumed to be in the landscape of interest and the landscape boundary represents an imaginary line around the entire grid. If the input image contains negatively valued cells, then those cells are assumed to be *outside* the landscape of interest and thus *outside* the landscape boundary. In this case, the edge between positively and negatively valued cells represents the landscape boundary. The landscape boundary is important in the absence of a landscape border (see below) because FRAGSTATS needs to know how to treat the edges along the boundary in all edge calculations. In the absence of a border, the landscape boundary will be treated according to user specifications (see below).

An image may include **background** (also referred to as ‘mask’)--an undefined area either ‘interior’ or ‘exterior’ to the landscape of interest. Note that background can exist as ‘holes’ in the landscape and/or can partially or completely surround the landscape of interest. The background value can be any integer value. Positively valued cells of background are assumed to be ‘inside’ the landscape of interest; negatively valued cells of background are assumed to be ‘outside’ the landscape of interest. This distinction is important, as noted above, because positively valued background (interior background) will be included in the total landscape area, and thus affect many metrics, even though it will not be treated as a patch per se (see below). Further, via the graphical user interface (see below), any class or combination of classes can be treated (i.e., reclassified) as background for a particular analysis. There are several critical issues regarding how background is handled by FRAGSTATS:

1. Interior background (i.e., positively valued background) is included in the total landscape area and thus affects metrics that include total landscape area in their calculations. However, and this is quite tricky, interior background is in essence excluded from the total landscape area in a number of class and landscape metrics that involve summarizing patch or class metrics. For example, mean patch area is based on the average size of patches at the class or landscape level. If interior background is present, mean patch size as computed by FRAGSTATS will not equal the total landscape area divided by the number of patches, because the total landscape area includes background area not accounted for in any patch. Similarly, the area-weighted mean of any patch metric (i.e., distribution statistics at the class and landscape level; see FRAGSTATS Metrics documentation)

weights each patch by its proportional area representation. Here, the proportional area of each patch is not based on the total landscape area, but rather the sum of all patch areas, which is equivalent to the total landscape area minus interior background. Similarly, a number of landscape metrics are computed from the proportion of the landscape in each class (e.g., Shannon's and Simpson's diversity). Here, proportional area of each class is not based on total landscape area because the proportions must sum to 1 across all classes. Instead, the proportions are based on the sum of all class areas, which is equivalent to the total landscape area minus interior background. Given the subtle differences in how interior background affects various metrics, it behooves you to carefully read the FRAGSTATS Metrics documentation pertaining to each metric you choose, assuming of course that interior background is an issue.

2. Exterior background (i.e., negatively valued background) essentially has no effect on the analysis or on the computation of any metrics. Exterior background is assumed to be 'outside' the landscape of interest. Thus, the extent of exterior background in the input landscape has no effect.
3. Background (both interior and exterior) cells adjacent to non-background classes represent edges that must be accounted for in all edge-related metrics. The user specifies how background edge segments should be handled in all edge-related calculations (see below).

An image also may include a **landscape border**; a strip of land surrounding the landscape of interest (i.e., outside the landscape boundary) within which patches have been delineated and classified. Patches in the border must be set to the negative of the appropriate patch type code. For example, if a border patch is a patch type of code 34, then its cell value must be -34. The border can be any width (as long as it is at least 1 cell wide) and provides information on patch type adjacency for patches on the edge of the landscape (i.e., along the landscape boundary). Essentially, patches in the border provide information on patch type adjacency for patches in the landscape of interest located along the landscape boundary; all other attributes of the patches in the border are ignored because they are *outside* the landscape of interest. Thus, the border affects only metrics where patch type adjacency is considered: core area, edge contrast, contagion and interspersed metrics.

Under most circumstances, it is probably not valid to assume that all edges function the same. Indeed, there is good evidence that edges vary in their effects on ecological processes and organisms depending on the nature of the edge (e.g., type of adjacent patches, degree of structural contrast, orientation, etc.). Accordingly, the user can specify a file containing **edge contrast weights** (described in more detail in the Contrast Metrics section of the FRAGSTATS Metrics documentation) for each combination of patch types (classes), including adjacencies involving background if it exists. Briefly, these weights represent the magnitude of edge contrast between adjacent patch types and must range between 0 (no contrast) and 1 (maximum contrast). Edge contrast weights are used to compute several edge-based metrics. If this weight file is not provided, these edge

contrast metrics are simply not computed. Generally, if a landscape border is designated, a weight file will be specified as well, because one of the principal reasons for specifying a border is when information on edge contrast is deemed important. If a border is present, the edge contrast associated with all landscape boundary edge segments is made explicit due to knowledge of the abutting patch types. If a border is absent, then all edge segments along the landscape boundary are treated the same as background, as specified in the user-provided edge contrast weight file. Note, however, that the presence of a landscape border will have no affect on the edge contrast metrics if a contrast weight file is not specified--because these metrics will not be computed.

Similarly, the user can specify a file containing **edge depths** (described in more detail in the Core Area Metrics section of the FRAGSTATS Metrics documentation) for each combination of patch types (classes), including adjacencies involving background if it exists. Briefly, edge depths represent the distance at which edge effects penetrate into a patch and must be given in distance units (meters); edge depths can be any number  $\geq 0$ . However, when implementing edge depths for the purpose of determining core areas, FRAGSTATS is constrained by minimum resolution established by the cell size. Thus, in effect, edge depths will be rounded to the nearest distance in increments of the cell size. For example, if the cell size is 30 m, and you specify a 100 m edge depth, the edge mask used to mask cells along the edge of a patch (i.e., eliminate them from the “core” of the patch) will be 3 cells wide (90 m), because it is not possible to use a mask that is 3.3 cells wide. Similarly, a specified edge depth of 50 m will in effect be rounded up to 2 cells (60 m). Therefore, it is generally advisable to specify edge depths in increments equal to the cell size. Edge depths are used to compute several core area-based metrics. If this edge depth file is not provided, these core area metrics are simply not computed. Typically, if a landscape border is designated, an edge depth file will be specified as well, because one of the principal reasons for specifying a border is when information on edge effects is deemed important. If a border is present, the edge depths associated with all landscape boundary edge segments is made explicit due to knowledge of the abutting patch types. If a border is absent, then all edge segments along the landscape boundary are treated the same as background, as specified in the user-provided edge depth file. Note, however, that the presence of a landscape border will have no affect on the core area metrics if an edge depth file is not specified--because these metrics will not be computed.

A landscape border is also useful for determining **patch type adjacency** for the contagion and interspersion indices. These metrics (described in more detail in the Contagion and Interspersion Metrics section of the FRAGSTATS Metrics documentation) require information on cell adjacency; that is, the abutting class values for the side of every cell. The proportional distribution of cell adjacencies is used to compute a variety of landscape texture metrics. Although a landscape border is not often designated for the primary purpose of computing these texture metrics, a border will inform the calculation of these metrics. If a border is present, the adjacencies associated with all landscape boundary edge segments is made explicit due to knowledge of the abutting patch types. If a border is absent, then all edge segments along the landscape boundary are treated the same as background and the corresponding cell adjacencies are ignored in the calculation of these metrics.

Metrics based on edge length (e.g., total edge or edge density) are affected by these considerations as well. If a landscape border is present, then edge segments along the boundary are evaluated to determine which segments represent ‘true’ edge and which do not. For example, an edge segment between cells with class value 5 (inside the landscape of interest) and cells with class value -5 (outside the landscape of interest; i.e., in the border) does not represent a *true edge*; in this case, the landscape boundary artificially bisects an otherwise contiguous patch and the edge is not counted in the calculations of total edge length. Conversely, an edge segment between class 5 and -3 represents a *true edge* and is counted. If a landscape border is absent, then a user-specified proportion of the landscape boundary is treated as true edge and the remainder is ignored. For example, if the user specifies that 50% of the landscape boundary should be treated as *true edge*, then 50% of the landscape boundary will be incorporated into the edge length metrics. Regardless of whether a landscape border is present or not, if a background class is specified, then a user-specified proportion of edge bordering background is treated as *true edge* and the remainder is ignored.

We recommend including a landscape border, especially if edge contrast, core area, or patch type adjacency is deemed important. In most cases, some portions of the landscape boundary will constitute ‘true’ edge (i.e., an edge with a contrast weight > 0) and others will not, and it will be difficult to estimate the proportion of the landscape boundary representing true edge. Moreover, it will be difficult to estimate the average edge contrast weight or edge depth for the entire landscape boundary. Thus, the decision on how to treat the landscape boundary will be somewhat subjective and may not accurately represent the landscape. In the absence of a landscape border, the affects of the decision regarding how to treat the landscape boundary on the landscape metrics will depend on landscape extent and heterogeneity. Larger and more heterogeneous landscapes will have a greater internal edge-to-boundary ratio and therefore the boundary will have less influence on the landscape metrics. Of course, only those metrics based on edge lengths and types are affected by the presence of a landscape border and the decision on how to treat the landscape boundary. When edge-based metrics are of particular importance to the investigation and the landscapes are small in extent and relatively homogeneous, the inclusion of a landscape border and the decision regarding the landscape boundary should be considered carefully.

So, let’s try to put all of this together. There are five types of metrics affected by landscape boundary, background, and border designations: (1) total landscape area, (2) edge length metrics, (3) core area metrics, (4) contrast metrics, and (5) contagion/interspersion metrics. Let’s consider several scenarios involving various combinations of background and border, and how each of these types of metrics will be treated under each scenario.

- Scenario 1.—Input landscape contains all positively valued cells of non-background classes (Fig. 1a). In this case, the entire grid is assumed to be in the landscape of interest and every cell belongs to a non-background class. The landscape boundary surrounds the entire grid and there is no border or background present.

*Total landscape area.*—All cells are included in the total landscape area calculation.

*Edge length metrics.*—User must specify the proportion of the landscape boundary to include as edge. All other edges are explicit.

*Core area metrics.*—The landscape boundary is treated like background; the user must specify the edge depth for cells abutting background in the edge depth file, and this depth is applied to the landscape boundary. All other edges are explicit; their edge depths are specified in the edge depth file.

*Contrast metrics.*—The landscape boundary is treated like background; the user must specify the edge contrast for cells abutting background in the edge contrast weight file, and this weight is applied to the landscape boundary. All other edges are explicit; their edge contrast weights are specified in the edge contrast weight file.

*Contagion/interspersion metrics.*—The landscape boundary is treated like background and is simply ignored since there is no information available on patch type adjacency.

- Scenario 2.—Input landscape contains all positively valued cells, but includes a background class (Fig. 1b). In this case, the entire grid is assumed to be in the landscape of interest, but some cells belong to a background class. Here, the background is *interior* because it is positively valued and thus *inside* the landscape of interest. The landscape boundary surrounds the entire grid and there is no border present.

*Total landscape area.*—All cells are included in the total landscape area calculation.

*Edge length metrics.*—User must specify the proportion of the landscape boundary and background edges to include as edge. All other edges are explicit.

*Core area metrics.*—The landscape boundary is treated like background; the user must specify the edge depth for cells abutting background in the edge depth file, and this depth is applied to both the landscape boundary and background edges. All other edges are explicit; their edge depths are specified in the edge depth file.

*Contrast metrics.*—The landscape boundary is treated like background; the user must specify the edge contrast for cells abutting background in the edge contrast weight file, and this weight is applied to both the landscape boundary and background edges. All other edges are explicit; their edge contrast weights are specified in the edge contrast weight file.

*Contagion/interspersions metrics.*—The landscape boundary and background are treated similarly; both are simply ignored when evaluating adjacencies since there is no information available on patch type adjacency in either case.

- Scenario 3.—Input landscape contains a mixture of positively valued cells and negatively valued background cells (Fig. 1c). Note, it doesn't matter whether the negatively valued background cells are located entirely on the periphery of the positively valued cells (i.e., outside the landscape of interest) or located as holes in the interior of the landscape, or a combination of the two. In all cases, the positively valued cells are assumed to be *inside* the landscape of interest, whereas the negatively valued background cells are assumed to be *outside* the landscape of interest and thus outside the landscape boundary. Here, the background is entirely *exterior* because it is all negatively valued and thus *outside* the landscape of interest. The landscape boundary separates contiguous regions of positively valued cells from the negatively valued cells and there is no border present. Alternatively, the exterior background could be considered border, but the use of border is generally reserved for situations involving negatively valued non-background cells.

*Total landscape area.*—All positively valued cells are included in the total landscape area calculation; negatively valued cells (here, all background) are ignored.

*Edge length metrics.*—User must specify the proportion of the landscape boundary and background edges (in this case, they are the same) to include as edge. All other edges are explicit.

*Core area metrics.*—The landscape boundary is treated like background; in this case, the entire landscape boundary is in fact also background. The user must specify the edge depth for cells abutting background in the edge depth file, and this depth is applied to the background edges (in this case, all on the landscape boundary). All other edges are explicit; their edge depths are specified in the edge depth file.

*Contrast metrics.*—The landscape boundary is treated like background; in this case, the entire landscape boundary is in fact also background. The user must specify the edge contrast for cells abutting background in the edge contrast weight file, and this weight is applied to the background edges (in this case, all on the landscape boundary). All other edges are explicit; their edge contrast weights are specified in the edge contrast weight file.

*Contagion/interspersions metrics.*—The landscape boundary and background (in this case, they are the same) are treated similarly; they are simply ignored when evaluating adjacencies since there is no information available on patch type adjacency.

- Scenario 4.—Input landscape contains a mixture of positively valued cells, including some positively valued background cells, and negatively valued background cells (Fig. 1d). Note, as in scenario 3, it doesn't matter whether the negatively valued background cells are located entirely on the periphery of the positively valued cells (i.e., outside the landscape of interest) or located as holes in the interior of the landscape, or a combination of the two. In all cases, the positively valued cells are assumed to be *inside* the landscape of interest, whereas the negatively valued background cells are assumed to be *outside* the landscape of interest and thus outside the landscape boundary. Here, the background is a combination of *interior* and *exterior* background. The landscape boundary separates contiguous regions of positively valued cells from the negatively valued cells and there is no border present. As noted in scenario 3, the exterior background could be considered border, but the use of border is generally reserved for situations involving negatively valued non-background cells.

*Total landscape area.*—All positively valued cells, including the 'interior' background, are included in the total landscape area calculation; negatively valued cells (here, all background) are ignored.

*Edge length metrics.*—User must specify the proportion of the landscape boundary (in this case, all background) and interior background edges to include as edge. All other edges are explicit.

*Core area metrics.*—The landscape boundary is treated like background; in this case, the entire landscape boundary is in fact also background. The user must specify the edge depth for cells abutting background in the edge depth file, and this depth is applied to all background edges (in this case, both on the landscape boundary and interior). All other edges are explicit; their edge depths are specified in the edge depth file.

*Contrast metrics.*—The landscape boundary is treated like background; in this case, the entire landscape boundary is in fact also background. The user must specify the edge contrast for cells abutting background in the edge contrast weight file, and this weight is applied to all background edges (in this case, both on the landscape boundary and interior). All other edges are explicit; their edge contrast weights are specified in the edge contrast weight file.

*Contagion/interspersion metrics.*—The landscape boundary (in this case, all background) and interior background are treated similarly; both are simply ignored when evaluating adjacencies since there is no information available on patch type adjacency in either case.

- Scenario 5.—Input landscape contains a mixture of positively valued non-background cells and negatively valued non-background cells (i.e., a true border; Fig. 1e). In this case, the positively valued cells are assumed to be *inside* the landscape of interest, whereas the negatively valued cells are assumed to be *outside* the landscape of

interest and thus outside the landscape boundary. The landscape boundary separates contiguous regions of positively valued cells from the negatively valued cells; no background exists; and there is a true border present. This is perhaps the *ideal scenario* because every cell is classified into a real class (i.e., no background) and a border is included to inform all edge, core, and adjacency calculations.

*Total landscape area.*—All positively valued cells are included in the total landscape area calculation; negatively valued cells are ignored.

*Edge length metrics.*—Because a border is present and there is no background, all edges are explicit; that is, the image provides explicit information on whether every edge segment along the boundary is a *true edge* or not. In this case, the user does not need to specify the proportion of the landscape boundary to include as edge. In fact, any user specification in this regard via the user interface will be disregarded.

*Core area metrics.*—Because a border is present and there is no background, all edges are explicit; that is, the image provides explicit information on the abutting patch types along the boundary. In this case, all edge depths are specified in the edge depth file.

*Contrast metrics.*—Because a border is present and there is no background, all edges are explicit; that is, the image provides explicit information on the abutting patch types along the boundary. In this case, all edge contrast weights are specified in the edge contrast weight file.

*Contagion/interspersion metrics.*—Because a border is present and there is no background, all edges are explicit; that is, the image provides explicit information on the abutting patch types along the boundary. In this case, all boundary edge segments are included in the adjacency calculations.

- Scenario 6.—Input landscape contains a mixture of positively valued cells, including both background and non-background classes, and negatively valued cells, including both background and non-background classes (Fig. 1f). This is the most complex scenario involving a complicated mixture in interior and exterior background and a border. In this case, all positively valued cells (including interior background) are assumed to be *inside* the landscape of interest, whereas the negatively valued cells are assumed to be *outside* the landscape of interest and thus outside the landscape boundary. The landscape boundary separates contiguous regions of positively valued cells from the negatively valued cells. A true border is present, but it includes some background class. This is perhaps also an ideal scenario, like scenario 5, but contains a realistic, sometimes unavoidable, situation in which some areas must be classified as background, either because there is no information available from which to classify them, or because it is deemed desirable ecologically to treat these areas as undefined background.

*Total landscape area.*—All positively valued cells, including the interior background, are included in the total landscape area calculation; negatively valued cells are ignored.

*Edge length metrics.*—Because a border is present but contains some background and there is interior background, only a portion of edges are explicit; that is, some edges abut background (either interior or exterior) and it is not explicit whether they represent *true edge* or not. In this case, the user must specify the proportion of edges involving background to include as edge.

*Core area metrics.*—Because a border is present but contains some background and there is interior background, only a portion of edges are explicit. Here, all boundary edges involving background and interior background edges are treated the same. The user must specify the edge depth for cells abutting background in the edge depth file, and this depth is applied to all background edges (in this case, both on the landscape boundary and interior). All other edges are explicit; their edge depths are specified in the edge depth file.

*Contrast metrics.*—Because a border is present but contains background, and there is interior background, only a portion of edges are explicit. Here, all boundary edges involving background and interior background edges are treated the same. The user must specify the edge contrast weight for cells abutting background in the edge contrast weight file, and this weight is applied to all background edges (both on the landscape boundary and interior). All other edges are explicit; their contrast weights are specified in the edge contrast weight file.

*Contagion/interspersion metrics.*—Because a border is present but contains some background, and there is interior background, only a portion of edges are explicit. In this case, edge segments involving background (both on the landscape boundary and interior) are ignored in the adjacency calculations.

## **Installation**

FRAGSTATS installation is quick and easy (hopefully). After downloading the fragstats.zip file, simply double click on the file and Winzip should open (if you don't have Winzip or another file compression utility that can unzip files, you will need to download Winzip from the web). In Winzip, simply click on the "install" button and follow the onscreen instructions. Alternatively, you can extract the zipped files to a folder and click on the **setup.exe** file and follow the instructions; however, this approach will not automatically delete the installation files after the installation is complete. Once installed, FRAGSTATS is run by double clicking your mouse on the **Fragstats.exe** file. However, if you are using FRAGSTATS with Arc Grids, then ArcView Spatial Analyst or ArcGIS must be installed on your computer and FRAGSTATS must have access to the dll libraries found in the ArcView Bin32 or ArcGIS Bin directory. Specifically, the Fragstats.exe file must either be located and run from the same directory as the dll libraries (not the best option) or a path to the dll libraries must be specified in your environmental variables (preferable) or in the autoexec.bat file. For detailed instructions

on installing Fragstats for use with ArcGrids, see the discussion on [Overview--Data Formats](#).

## Data Formats

FRAGSTATS accepts raster images in a variety of formats, including ArcGrid, ASCII, BINARY, ERDAS, and IDRISI image files. FRAGSTATS does not accept Arc/Info vector coverages like the earlier version 2. All input data formats have the following common requirements:

- All input grids should be *signed integer* grids containing all **non-zero** class values (i.e., each cell should be assigned an integer value corresponding to its class membership or patch type). Note, FRAGSTATS assumes that the grids are ‘signed’ integer grids; inputting an ‘unsigned’ integer grid may cause problems. In addition, note that assigning the zero value to a class is allowable, but can cause problems in the moving window analysis if it is not specified as the background class. Specifically, if zero is specified as the background class, FRAGSTATS will reclassify all zero cell values to a new class value equal to 1 plus the largest class value in the input landscape. This procedure is necessary because a zero background class value will cause problems in the moving window analysis because a background value of zero in the output grid cannot be distinguished from a computed metric value of zero. In addition, a zero class value will cause problems when the landscape contains a border because zero cannot be negative, yet all cells in the border must be negative integers). Thus, all zero cells are assumed to be interior (i.e., inside the landscape of interest). If a moving window analysis is selected and the input grid contains zeros and zero is NOT specified as the background class, then execution will be stopped and an error message reported to the log file. For these reasons, it is best to avoid the use of zero class values altogether.
- All input grids should consist of *square* cells with cell size specified in *meters*. For certain input formats (ASCII and BINARY), this is not an issue because cells are assumed to be square and you are required to enter the cell size (in meters) in the graphical user interface. However, FRAGSTATS assumes that all other image formats (ArcGrid, ERDAS, and IDRISI) include header information that defines cell size. Consequently these images must have a metric projection (e.g., UTM) to ensure that cell size is given in metric units.

There are some additional special considerations for each input data format, as follows:

(1) **Arc Grid** created with Arc/Info. Note, to use Arc Grids you must have ArcView Spatial Analyst or ArcGIS installed on your computer and FRAGSTATS must have access to a certain .dll file found either in the ArcView Bin32 directory (for ArcView Spatial Analyst users) or the ArcGIS Bin directory (for ArcGIS users). Specifically, a path to the corresponding dll library file should be specified in the environmental settings under NT or Windows 2000 operating systems, or a path statement included in the autoexec.bat file, e.g., under Windows 98, as follows:

Windows NT: You can add the necessary Path variable or edit the existing one via the Control panel - System Properties - Environment tab. Add a new variable or edit the existing Path variable in the system variables, not the user variables (this will require administrative privileges). Add the full path to the appropriate .dll file. If you are using ArcView Spatial Analyst, the required file is the avgridio.dll file and it is typically installed in the following path: \esri\av\_gis30\arcview\bin32. If you are using ArcGIS, the required file is the agridio.dll file and it is typically installed in the following path: \esri\arcinfo\arcexe81\bin. Note, your software version number and path may be different so be sure to locate the .dll file on your computer and enter the correct path. If you are using both Spatial Analyst and ArcGIS, then you can enter either or both paths to the Path system variable.

Windows 2000/XP: You can add the necessary Path variable or edit the existing one via the Control panel - System Properties - Advanced tab - Environment Variables button. Add a new variable or edit the existing Path variable in the system variables, not the user variables (this will require administrative privileges). Then, following the instructions above for Windows NT.

Windows 95/98: You must add the necessary Path statement to the autoexec.bat file. First, search your computer for the autoexec.bat file and open it using any text editor. Then, either add a Path statement or edit the existing one. Add the full path to the appropriate .dll file. If you are using ArcView Spatial Analyst, the required file is the avgridio.dll file and it is typically installed in the following path: \esri\av\_gis30\arcview\bin32. If you are using ArcGIS, the required file is the agridio.dll file and it is typically installed in the following path: \esri\arcinfo\arcexe81\bin. Thus, the path statement should look something like: PATH c:\esri\av\_gis30\arcview\bin32. Note, your software version number and path may be different so be sure to locate the .dll file on your computer and enter the correct path. If you are using both Spatial Analyst and ArcGIS, then you can enter either or both paths to the Path system variable. If you are adding the path to an existing path, simply use a semicolon to separate the unique paths in the Path statement. After saving the file you will need to reboot your machine for the change to take effect.

(2) **ASCII** file, no header. Each record should contain 1 image row. Cell values should be separated by a comma or a space(s). Note, it will be necessary to strip (delete) the header information from the image file if it exists, but be sure to keep it for later reference regarding background cell value, # rows, and # columns.

(3) **32-bit binary** file, no header; no other limitations.

(4) **16-bit binary** file, no header. Patch ID output file, if selected, will be output in *signed 32-bit integer* format to accommodate a greater number of patches. In addition, because moving window analysis requires floating points, if moving window analysis is selected, the output grids produced will be *32-bit floating point* grids.

(5) **8-bit binary** file, no header. Patch ID output file, if selected, will be output in *signed 32-bit integer* format to accommodate a greater number of patches. In addition, because moving window analysis requires floating points, if moving window analysis is selected, the output grids produced will be *32-bit floating point* grids.

(5) **ERDAS** image files (.gis, .lan, and .img). FRAGSTATS accepts images from both ERDAS 7 (.gis and .lan) and ERDAS 8 (.gis, .lan, and .img), but the limitations are somewhat different, as follows:

ERDAS 8 Files.—FRAGSTATS accepts .gis, .lan, and .img files used by current versions of ERDAS IMAGINE, including *signed 8-, 16-, and 32-bit integer* grids. While .gis and .lan file formats are supported, their limitations make them less practical than .img (see discussion of ERDAS 7 files below). Care should be taken when preparing the data to be used with FRAGSTATS, especially when importing data from other formats (for instance from Arc Grid). Be sure to set the import options to '*signed integer*'. Regardless of the input integer format (8-, 16-, or 32-bit), the patch ID output file created by FRAGSTATS will be a *signed 32-bit integer*, and if moving window analysis is selected, all output grids will be *32-bit floating point*. Multi-layered files are not rejected, but only layer one is processed and the outputs are all single layered. As noted above, cells must be square, not rectangular, and the measurement unit should be meters. If the measurement unit is not specified, then it is assumed to be meters. The cell size and measurement unit specification can be changed within ERDAS Imagine using the ImageInfo tool -> Change Map Model from the Edit menu. The projection information existing in the input file is passed unchanged to the output files. FRAGSTATS does not use this information internally.

ERDAS 7 files.-- FRAGSTATS accepts .gis and .lan files used by ERDAS 7, which are limited to *unsigned 8- or 16-bit integer* grids. FRAGSTATS will accept both *8-bit* and *16-bit integer* files and it will reject multi-layered files. While .gis and .lan file formats are supported, their limitations make them less practical than .img. In particular, ERDAS 7.x files are limited to *unsigned integers* (i.e., only positive integers), therefore landscape borders (which require negative class values) cannot be represented. Another consequence of this particular limitation is that FRAGSTATS-generated patch ID files will not fill the non-landscape cells (i.e., no data cells) with the usual value (minus the background class value), but with zero values. The restriction to 8- or 16-bit integers imposes some limitations when using the FRAGSTATS on large grids. Specifically, unsigned 16-bit integers can only take on values up to 65,534. Thus, class ID values are limited to integers within this range (note, this should not be a problem, since it is unlikely that anyone would have more than 65,534 patch types). Similarly, patch ID values in the patch ID output grid optionally produced by FRAGSTATS are limited to the same range, effectively limiting the number of patches in this output grid to 65,534. If the grid contains more than this number of patches, FRAGSTATS will not be able to output a unique ID for each patch and the user will have to somehow distinguish among patches with the same ID. Finally, because the moving window analysis requires 32-bit output files (in

order to output floating point values), moving window analysis is not supported with ERDAS 7 files.

(6) **IDRISI** image files (.rdc). IDRISI currently supports *signed 8- or 16-bit integers* and *32-bit floating point grids*. This imposes some limitations when using the FRAGSTATS on large grids. Specifically, signed 16-bit integers can only take on values between -32,768 and 32,767. Thus, class ID values are limited to integers within this range (note, this should not be a problem, since it is unlikely that anyone would have more than 32,767 patch types). Similarly, patch ID values in the patch ID output grid optionally produced by FRAGSTATS are limited to the same range, effectively limiting the number of patches in this output grid to 37,767. If the grid contains more than this number of patches, FRAGSTATS will not be able to output a unique ID for each patch and the user will have to somehow distinguish among patches with the same ID. Fortunately, IDRISI supports 32-bit floating point grids, which are produced in a moving window analysis.

### Step 1. Starting the FRAGSTATS GUI

To start FRAGSTATS, simply double click on the **Fragstats.exe** file and (hopefully) the opening FRAGSTATS window will display (Fig. 2).

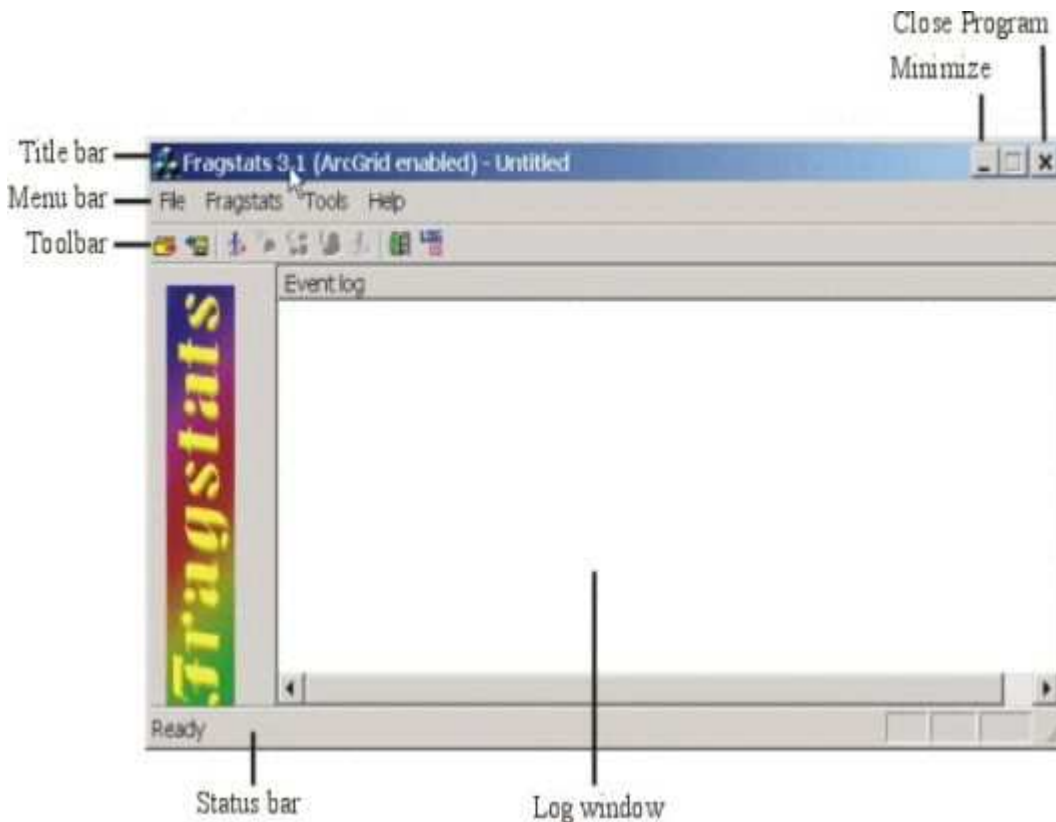


Figure 2. Anatomy of the FRAGSTATS opening user interface.

The anatomy of this opening window is quite simple, as it is similar to most windows-based programs. The *title bar* lists the name of the current or open parameterization file. The parameterization file contains the current parameterization scheme. When you first

start FRAGSTATS, a parameterization file does not exist. Consequently, the title is listed as “untitled” until you save the current parameterization scheme under the **File-Save** option. Once you have saved the current parameterization or have retrieved a previously saved scheme, the title bar will list the open parameterization file. The *menu bar* consists of a few items that allow you to parameterize a FRAGSTATS run, edit certain files, and save and retrieve parameterization files. Each menu item has a drop-down list of options. The *toolbar* provides quick access to the frequently used **File** and **Fragstats** menu options. The *log window* provides a running log of the analysis. The *status bar* indicates the status of the program. Details of the parameterization process are described in Step 2. Here, each of the menu bar items are briefly described:

**File Menu.**—The File menu allows you to open an existing parameterization file or save the current parameterization file. Note, parameterization files are given the file extension .frg.

1. *New.*—Creates a new (or blank) parameterization file. It is not necessary to begin by creating a new file. A new file is automatically opened when you start FRAGSTATS. This option is used when you have a parameterization file currently open and wish to create a new file from scratch.
2. *Open.*—Opens an existing (previously saved) parameterization file.
3. *Save.*—Saves the current parameterization to a file with the extension .frg. Note, if you are saving a parameterization file for the first time, you will be prompted to specify a location and file name. If you are saving from an open parameterization file, this will overwrite the existing file without prompting you for a file name. This file contains all the parameter settings in the dialog boxes at the time the file was saved. This can be very useful if you are running FRAGSTATS repeatedly with the same or similar parameterization schemes.
4. *Save as.*—Saves the current parameterization file to a location and file name that you specify.
5. *Exit.*—Closes the program. The same effect is achieved by clicking on the X in the upper right corner of the window.

**Fragstats Menu.**—The Fragstats menu allows you to parameterize the analysis.

1. *Set Run Parameters.*—Opens a new dialog box that allows you to specify necessary parameters for running FRAGSTATS (see Step 2).
2. *Select Patch Metrics.*—Opens a new dialog box that allows you to select and parameterize the desired patch-level metrics. Note, this function is not active until the Run Parameters dialog box is successfully completed, and then only if patch metrics are selected in the run parameters.

3. *Select Class Metrics.*—Opens a new dialog box that allows you to select and parameterize the desired class-level metrics. Note, this function is not active until the Run Parameters dialog box is successfully completed, and then only if class metrics are selected in the run parameters.
4. *Select Landscape Metrics.*—Opens a new dialog box that allows you to select and parameterize the desired landscape-level metrics. Note, this function is not active until the Run Parameters dialog box is successfully completed, and then only if landscape metrics are selected in the run parameters.
5. *Clear All Menus.*—Clears the Run Parameters, Patch-, Class-, and Landscape Metrics dialog boxes.
6. *Execute Fragstats.*—Executes a FRAGSTATS run. An “execution finished” message will appear in the log window when FRAGSTATS processing is complete.

**Tools Menu.**—The Tools menu provides you with several miscellaneous options for creating and editing batch files, reclassifying classes and controlling class-level output, browsing results, and saving the log file.

1. *Batch File Editor.*—Allows you to create and edit batch files. See Step 4 on Using Batch Files for details on the batch file editor.
2. *Class Properties.*—Allows you to edit the properties of each class; that is, designate classes as background and disable the output for selected classes. See Step 5 on Modifying Class Properties for details.
3. *Browse Results.*—Allows you to browse and save the results of the analysis. See Step 7 on Browsing and Saving Results for details.
4. *Clear Log.*—Clears the log window of all text.
5. *Save Log.*—Saves the log window as an ASCII text file with the extension .log to a location and file name that you specify.

**Help Menu.**—The Help menu provides you with online help.

1. *Help Content.*—Allows you to access a familiar windows-based help interface with the standard content, index, and find options. The online help information is essentially the same as provided in this document and the FRAGSTATS Metrics document.
2. *About Fragstats.*—Lists the authors and the software release version.

## Step 2. Setting Run Parameters

After opening the FRAGSTATS interface, the next step is to set the run parameters. By selecting the set run parameters option in the **Fragstats** drop-down menu, the Run Parameters dialog box will display (Fig. 3).

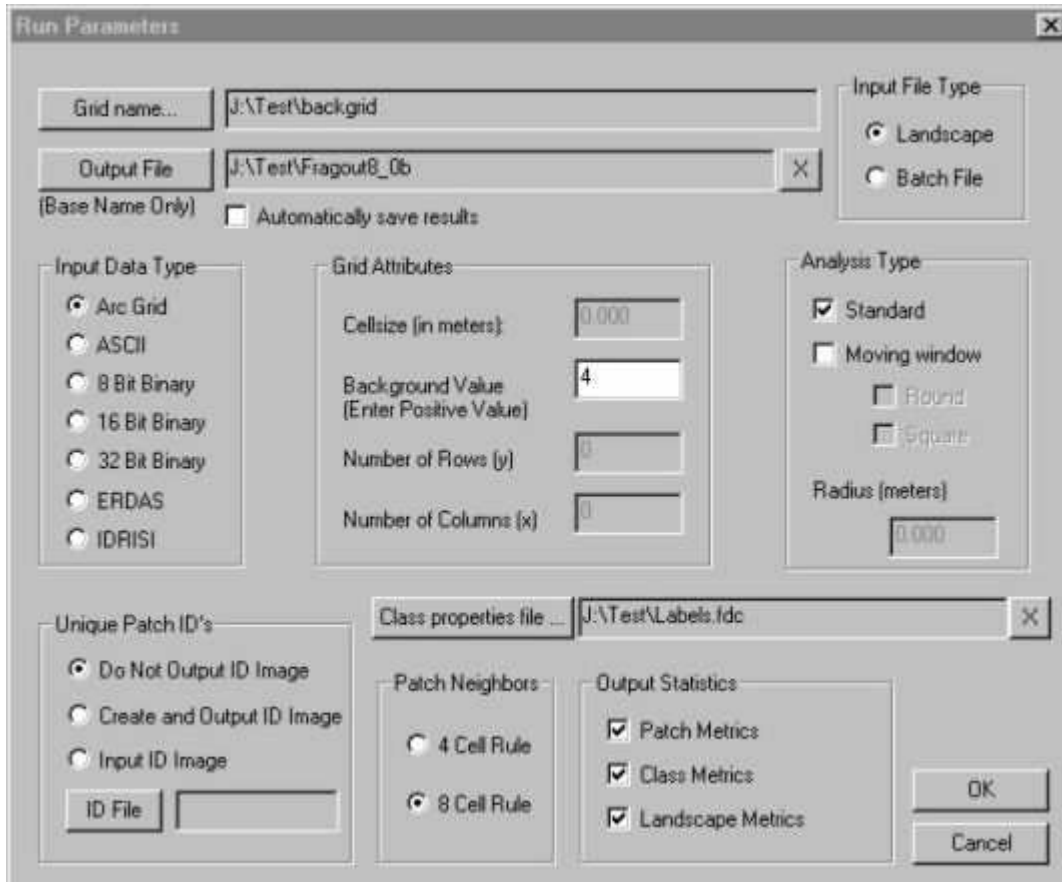


Figure 3. The Run Parameters dialog box.

This dialog box contains all the essential input parameters required to run FRAGSTATS except for the selection of the metrics themselves. Here is where you define the input grid or batch file, specify the input data type, choose an analysis type, select the patch ID output option, choose a neighbor rule for patch delineation, specify a class properties file, and select the levels of metrics you want computed. This contains most of the information contained in the command line (or most of the prompts) in version 2. The order in which the run parameters are specified is not important, although there is a logical sequence for at least a few of the parameters. The parameters are stored only after the OK button is clicked.

1. *Input File Type*.—Chose between Landscape and Batch modes:

(1) **Landscape.**--If Landscape mode is selected, then FRAGSTATS expects the designated input file to be a single raster image; FRAGSTATS will produce the conventional output for that landscape.

(2) **Batch File.**--If Batch mode is selected, then FRAGSTATS will run the batch file specified in the Input File text box and produce output for all of the landscapes designated in the batch file. See Step 4 on Working with Batch Files for details on building a suitable batch file. If Batch mode is selected, then the Input Data Type (see below) and Grid Attributes (see below) will become inactive (grayed out in the interface) because these parameters will be specified uniquely for each input landscape in the batch file.

2. ***Input Data Type.***--FRAGSTATS accepts several types of input image data formats. See [Data Formats](#) in the Overview section for details. Briefly, all input images should be integer grids with *non-zero* class values (i.e., each cell should be assigned an integer value corresponding to its class membership or patch type). Note, assigning the value 0 to a non-background class is not allowed in the moving window analysis, and an error message will be reported to the log file in this case. In addition, all input grids should consist of *square* cells with the measurement units in *meters*. Choose one of the following:

(1) **Arc Grid** created with Arc/Info.

(2) **ASCII** file, no header.

(3) **8 bit binary** file, no header.

(4) **16 bit binary** file, no header.

(5) **32 bit binary** file, no header.

(6) **ERDAS** image files (ERDAS 7 .gis or .lan, 8 or 16 bit files; and ERDAS 8 .img, 8, 16, or 32 bit files).

(7) **IDRISI** image files (.rdc).

3. ***Input File Name.***--Specify an input file, where the input file is either the input image to be analyzed (landscape mode) or a batch file, by clicking on the file name button and navigating to the desired input file. If the Input Data Type is selected first, then navigation to the input file will default to files of the specified type. For example, if Arc Grid is selected as the Input Data Type, then the navigation window will reveal only Arc Grids. In addition, the label on the File Name button will reflect the Input Data Type. Once you have selected a file, the full path and file name will be displayed in the text box next to the File Name button.

4. Output File.--Specify a “basename” for the output files by clicking on the Output File button and navigating to the desired folder, and then entering a “basename” or selecting an existing file to overwrite. This basename will be given the extensions .patch, .class, .land and .adj for the corresponding patch, class, and landscape metrics and adjacency matrix. Note, you can automatically save the results to output files by checking the Automatically Save Results check box. In addition, it is not necessary to enter an output file basename here. The results can also be saved from the **browse** dialog box (see Step 7 on Browsing and Saving Results). However, if an output file name is specified here, then it will be used as the default basename when saving the results from the **browse** dialog box. If an output file name is not specified here, then you will be prompted to provide a basename later when saving the results. Note, if you specify an output file name that already exists, FRAGSTATS will prompt you as to whether you want to overwrite the existing files after execution of the program is complete.

5. Analysis Type.--Chose between Standard and Moving Window modes:

(1) **Standard**.--If Standard mode is selected, then FRAGSTATS will produce the conventional output for the input landscape(s) consisting of the .patch, .class, and .land files corresponding to the patch, class, and landscape metrics.

(2) **Moving Window**.--If Moving Window mode is selected, then FRAGSTATS will conduct a moving window analysis and output a new grid for each selected metric. In addition, if you select a moving window analysis, then you must specify the shape (round or square) and size (radius, in meters) of the window to be used. A window of the specified shape and size is passed over every positively valued cell in the grid (i.e., all cells *inside* the landscape of interest). However, only cells in which the entire window is contained within the landscape are evaluated (see Boundary Effects below). Within each window, each selected metric at the class or landscape level is computed and the value returned to the focal (center) cell. Patch metrics are not allowed in the moving window analysis. The moving window is passed over the grid until every positively valued cell (including positively valued background cells) containing a *full* window is assessed in this manner. Note, internal background cells containing real positively-valued classes in the window may receive a value in the output grid, despite the fact that the cell is background in the input grid. Specifically, if the entire window is internal background, then the cell will receive a minus background value in the output grid. There are several important considerations when using the moving window analysis type:

- Window Shape and Size.--The user-specified window size refers to the radius (in meters) of a near-circular window or square window, depending on the shape chosen. If a square window is selected, the radius refers to the shortest distance between the focal cell and the side of the square window; hence, it refers to the distance from the focal cell to the side of the square in an orthogonal direction. The actual area of the window as implemented algorithmically will vary slightly from the area calculated mathematically based on the geometry of a circle or

square for two reasons. First, the radius is given as the distance from the focal cell to the edge of the window. For example, given a cell size of 10 m and a circular window, a 40 m radius would be implemented as a mask 4 cells wide. The diameter of the window would equal 90 m—twice the radius plus the size of the focal cell—as opposed to 80 m. The addition of the focal cell to the diameter of the window is necessary to force the focal cell to always be located at the exact center of the window. Second, the specified radius (in meters) is always rounded to the nearest cell. Thus, if the radius is not perfectly divisible by the cell size, the actual window will be somewhat smaller or larger.

- Boundary Effects.—Cells located close to the edge of the landscape (i.e., near the landscape boundary) will be biased in moving window calculations if the window intersects the landscape boundary. Consider a cell located on the landscape boundary. A normal window placed on that cell would extend well outside the landscape boundary, in fact, half the window would extend beyond the landscape where information on landscape structure is absent. In fact, any cell within the specified radius of a round window or  $\frac{1}{2}$  the length of a side of a square window will be biased in this way. There are several alternative ways of handling this bias. FRAGSTATS adopts a conservative method that involves simply not computing the metrics for focal cells containing a *partial* window (i.e., a window not fully contained within the input grid). Actually, FRAGSTATS returns the user-specified exterior (negative) background value for these cells. Thus, in practice, the output grid will contain a peripheral buffer of exterior background cells surrounding the *core* of the landscape—only the core (cells containing a *full* window) will contain metric values (Fig. 4a). Clearly, as landscape extent increases relative to window size, the magnitude and spatial extent of this boundary issue decreases. For this reason, care should be exercised in selecting a window size that minimizes the loss of information due to this boundary effect. An alternative approach for dealing with this boundary effect is to expand the extent of the input landscape to include a suitably wide **expansion strip** of positively valued and appropriately classified cells around the actual landscape of interest, where the width of this extension is equal to the radius of the window (Fig. 4b). In this manner, the *core* of the landscape in the output grid produced by the moving window analysis will align with the original landscape boundary of interest. It is important to realize that including a suitably wide **landscape border** (negatively valued, but classified cells) does not have the same effect. Border, by definition, consists of negatively valued cells outside the landscape of interest, and FRAGSTATS ignores all negatively valued cells when calculating metrics, except for the information provided on adjacency to positively valued cells. Thus, a window that includes cells in the landscape border, will effectively function like a ‘partial’ window, and therefore bias the calculation of all metrics.

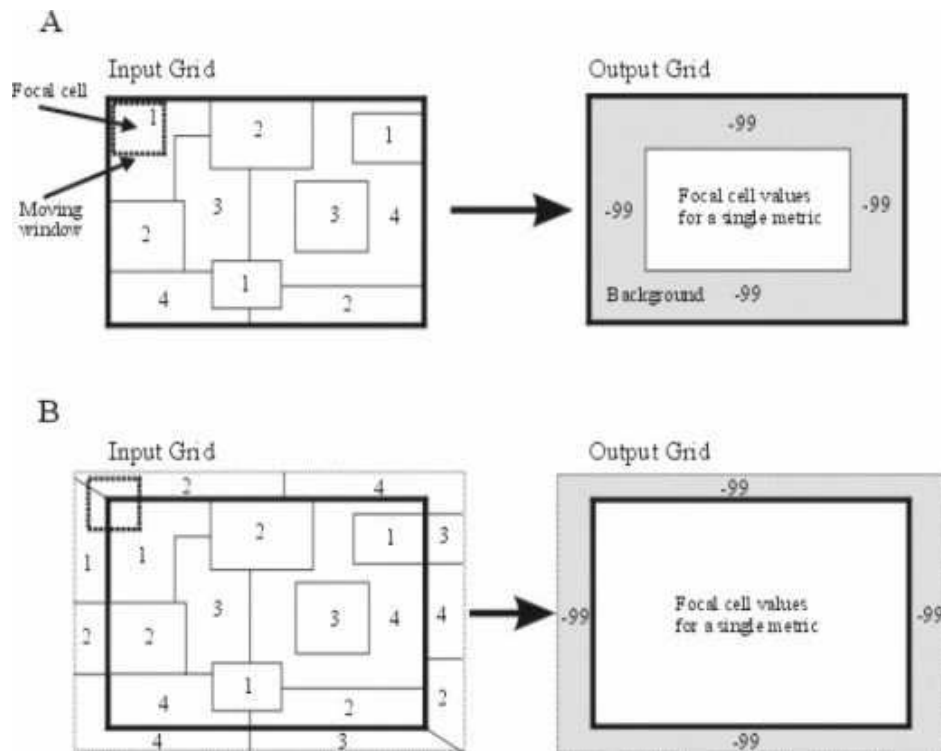


Figure (A)

4.

Moving window applied to an input grid without a border produces an output grid for each unique class-metric combination in which a strip the width of the window around the periphery of the grid is given a background value in the output grid. (B) A landscape border at least as wide as the window allows all cells inside the landscape boundary (dark line) (i.e., positive values) to be given the computed focal cell value.

- Input Data Types.—Moving window analysis is restricted to input data types that can effectively handle floating point values. Thus, the 8- and 16-bit binary data formats are not allowed in a moving window analysis and will be inactive in the Run Parameters dialog box if the moving window analysis type is selected. If this data type is included in a batch file used in conjunction with a moving window analysis, the corresponding records will be ignored.
- Selection of Classes and Metrics.—For each selected metric and enabled class, FRAGSTATS outputs a separate grid (in the same format as the input grid). Thus, if the input landscape contains 10 classes and all of them are *enabled* in the class properties file (see Step 5 on Modifying Class Properties) and you select 1 class level metric, then FRAGSTATS outputs 10 grids, one for each class for the selected metric. In this case, each output grid represents a separate class, and the cell values represent the computed values of the selected metric for that class. Specifically, a window is placed over the first cell in the input landscape, the selected metric is computed for the first class, and this value is output to the corresponding cell in a new grid for that specific metric-class combination. The process is repeated for the next class, and so on, until all classes have been assessed. Next, the window is placed over the next cell and the process is repeated. The process is repeated in this manner until all positively valued cells

containing a full window in the input landscape have been evaluated. The end result is a new grid for each class, in which the cell values represent the values of the computed metric. Accordingly, if you select 5 class-level metrics, then FRAGSTATS outputs 50 grids, one for each class-metric combination. If, in addition to these class metrics, you also select 3 landscape metrics, FRAGSTATS outputs an additional 3 grids, one for each landscape metric. Clearly, the number of output grids can increase quickly with several classes and several metrics. Thus, it is important to carefully select the most parsimonious set of classes and metrics. Note, windows containing no cells of the corresponding class, or in some cases just a single cell of the corresponding class, will be assigned a background value in the output grid.

- Computer Processing and Memory Requirements.—The computer processing and memory demands of the moving window analysis are phenomenal. Consider a relatively small grid of 100 x 100 cells; i.e., 10,000 cells. The moving window analysis involves placing a window over every cell and computing one or more metrics. This is equivalent to doing 10,000 FRAGSTATS analyses. Now imagine that you have a larger grid of 1000 x 1000 cells; i.e., 1,000,000 cells. Clearly, the processing time quickly becomes overwhelming. In addition, the memory demands increase as a function of the size the input grid. FRAGSTATS must be able to allocate memory for at three grids, where each grid requires 4 bytes for every cell. See *Computer Requirements* in the Overview Section for a detailed description of the memory requirements. Clearly, given the limited memory available in most personal computers, it is quite possible that you will not have enough memory to do even a single unique class-metric combination, let alone the dozens or hundreds that could easily result if you selected several classes and several metrics. If more than one class-metric combinations are selected, FRAGSTATS will determine how many can be done given the available memory and then parse the job into separate passes. For example, if 20 class-metric combinations are selected, but available memory is sufficient for only 4 at a time, then FRAGSTATS will conduct 5 passes across the landscape, output 4 grids each pass. Given these considerations, it behooves you to use this option sparingly and with great patience until computer processing capabilities increase substantially. And don't be too surprised if your computer is simply unable to allocate sufficient memory to do any moving window analysis.
- Output Grid Naming Convention.—Given the number of possible output grids produced from a moving window analysis and limits on the file name length with some data types (e.g., Arc Grids), the output file naming convention is somewhat cumbersome. If a moving window analysis is selected, FRAGSTATS will create a new subdirectory beneath the directory containing the input file by appending “\_MW1” (for moving window #1) to the name of the input file. Thus, a directory named “Test” containing the input file named “TestGrid” will have a new subdirectory under the Test directory named “TestGrid\_MW1”. This subdirectory will contain an output grid for each class-metric combination and landscape metric selected. For landscape metrics, the output grids are named using the

metric acronym. For class metrics, the metric acronym is combined with the class ID value (see Step 5 on Modifying Class Properties) because each class has a separate output grid. For example, the landscape-level contagion metric (CONTAG) would be given the following grid name: CONTAG. The class-level clumpiness metric for class ID #3 would be given the following grid name: CLUMPY\_3. See the list of metrics in the FRAGSTATS Metrics documentation for the metric acronyms. If a second moving window analysis is conducted on the same input file (e.g., using a different window size), a second directory is created by appending “\_MW2” to the name of the input file. And so on for each subsequent moving window analysis.

- Grid Attributes.—Depending on Input Data Type, you will need to enter some information about the grid. Note, only the text boxes that are required for the corresponding Input Data Type will be active; all others will be grayed out. Fill in all text boxes that are active.

(1) **Cell Size** (in meters).—Enter the size of cells in meters in the input image. Cells must be square. The length of 1 side of a cell should be input.

(2) **Background Value**.—[Optional] Enter the value to be used for background cells. This is only required if there are cells interior or exterior to the landscape of interest that you want to treat as background (see Overview discussion). Note, it is possible to specify multiple class values as background, but this must be done from the **Class Properties** tool under the **Tools** menu. When this is done, the designated classes are reclassified to the background value specified here in the Grid Attributes. Note, all background cells are assigned this cell value, and this can have important implications if you select core area or edge contrast metrics. Specifically, if you wish to specify a non-zero edge depth or edge contrast weight to background edges, you must include this background class value in the pairwise combination of classes given in the edge depth and contrast weight files (see below).

(3) **Number of Rows**.—Enter the number of rows in the input image. This is only required if Input Data Type is ASCII or Binary.

(4) **Number of Columns**.—Enter the number of columns in the input image. This is only required if Input Data Type is ASCII or Binary.

- Unique Patch Ids.—Chose among the following three options for outputting a Patch ID image:

(1) **Do Not Output ID Image**.—A patch ID image will not be output.

(2) **Create and Output ID Image**.—A patch ID image will be created by FRAGSTATS and output in the same data type format as the Input Data Type. However, all binary input data formats as well as any ERDAS 8 input data format will be output in *signed 32-bit integer* format to accommodate a greater number of

patches. The Patch ID image will contain a unique ID for each patch in the landscape. All background cells will be assigned a negative of the user-specified background value. This patch ID corresponds to the patch ID in the “basename”.patch output file. This image is needed if you wish to associate the patch-level output with specific patches. Note, the patch ID file will be named using the following convention:

Input file name + 4 or 8 (depending on neighbor rule) + ID

Thus, an input file named “test” analyzed with an 8-neighbor rule will be given the following patch ID file name: test8id. If you attempt to create and output an ID image that already exists, e.g., from a previous run, FRAGSTATS will ask you whether you want to overwrite the existing file. NOTE, if you are using Arc Grids and if you attempt to create and output an ID image with the same name as a grid that is currently open in another program, e.g., in ArcView, the grid will be corrupted and an error message will be written to the log window. In this case, the grid folder must be deleted, even after closing ArcView, before you can create and output an ID image with that name.

(3) **Input Unique ID Image.**—If you select this option, you must specify an existing patch ID image. The patch IDs on this image will correspond to the patch IDs in the “basename”.patch output file. Also, the data type of this file must be the same as the Input Data Type.

- ***Patch Neighbor Rule.***—Chose between the 4-cell and 8-cell rule. The 4-cell rule considers only the 4 adjacent cells that share a side with the focal cell (i.e., orthogonal neighbors) for determining patch membership. The 8-cell rule considers all 8 adjacent cells, including the 4 orthogonal and 4 diagonal neighbors. Thus, if the 4-cell rule is selected, two cells of the same class that are diagonally touching will be considered as part of separate patches; if the 8-cell rule is selected, these will be considered part of the same patch. The choice of patch neighbor rule will affect most of the configuration metrics, but will have no affect on the composition metrics.
- ***Class Properties File.***—Specify a Class Properties File by clicking on the corresponding button and navigating to the desired file. Note, FRAGSTATS uses the file extension .fdc for class properties files and will look for files with this extension by default when navigating. The .fdc extension is not mandatory, but using it can help keep files organized. Each record in the file should contain a numeric patch type value, the character descriptor for that patch type, a status indicator, and a background indicator. The syntax for this *comma-delimited ASCII file* is as follows:

*ClassID, ClassName, Status, isBackground*

- *ClassID* is an integer value corresponding to a class value in the landscape.
- *ClassName* is a descriptive name of the class; descriptive names can be any length and contain any characters, including spaces, but cannot include

commas. This descriptive name is reported in all patch and class output files for the variable TYPE.

- *Status* can take on the values: true, or t; and false, or f., and determines whether the corresponding class should be processed and added to the results or simply ignored in the output files. A “true” or “t” indicates that the class is *enabled* and should be output in the patch and class output files. A “false” or “f” indicates that the class is *disabled* and should not be output. Note, class status does not effect the computation of landscape metrics; disabled classes are still included, as necessary, in the computation of landscape metrics. Although there is some savings of computer processing by disabling a class, the primary effect is on the output. This feature allows you to “turn off” classes that you are not interested in so that you don’t have to view their statistics in the output files.
- *isBackground* can take on the values: true, or t; and false, or f, and determines whether the corresponding class should be reclassified and treated as background (i.e., assigned the background value specified in the Grid Attributes. Note, classifying a class as background will have an effect on many landscape metrics (see Overview discussion).

The class properties file should contain a record for each class in the input landscape, and all arguments should be separated by a comma or space(s). For example:

```
1,shrubs,true,false  
2,conifers,true, false  
3,deciduous,true,false  
4,other,false,true  
etc.
```

In summary, the class properties file allows you to do three things: (1) specify character descriptors for each class in order to facilitate interpretation of the output files, (2) limit the output files to only the classes of interest, and (3) reclassify classes to background.

NOTE, if the class properties file is provided, the class names will be written to the output files. Otherwise, the class IDs (numeric patch type codes) will be written to the output files. In addition, if the class properties file is not provided, by default all classes in the input landscape will have class-level metrics computed and output to the class output file--assuming that class metrics are selected. In addition, if you do not specify a class properties file, then all cells that you wish to treat as background must have the background value specified in the Run Parameters dialog box, because there is no way to reclassify classes as background in FRAGSTATS without a class properties file.

- Output Statistics.—Select the levels of metrics you want computed:

(1) **Patch Metrics.**—If selected, patch metrics can be computed. However, individual patch metrics must be selected in the **Patch Metrics** dialog box (see Step 3), otherwise no patch metrics will be calculated.

(2) **Class Metrics.**—If selected, class metrics can be computed. However, individual class metrics must be selected in the **Class Metrics** dialog box (see Step 3), otherwise no class metrics will be calculated.

(3) **Landscape Metrics.**—If selected, landscape metrics can be computed. However, individual landscape metrics must be selected in the **Landscape Metrics** dialog box (see Step 3), otherwise no landscape metrics will be calculated.

### **Step 3. Selecting and Parameterizing Patch, Class, and Landscape Metrics**

The next step is to select and parameterize the patch, class, and landscape metrics. Note, these options are available only if you select the respective output statistics in the Run Parameters dialog box. By selecting the select patch metrics option in the **Fragstats** drop-down menu, the **Patch Metrics** dialog box will display (Fig. 5). Similarly, by selecting the select class metrics or select landscape metrics options, the corresponding dialog boxes will display. Each dialog box consists of a series of tabbed pages. Each tabbed page represents a group of related metrics. The organization of these metric groups is discussed in the Background Material documentation.

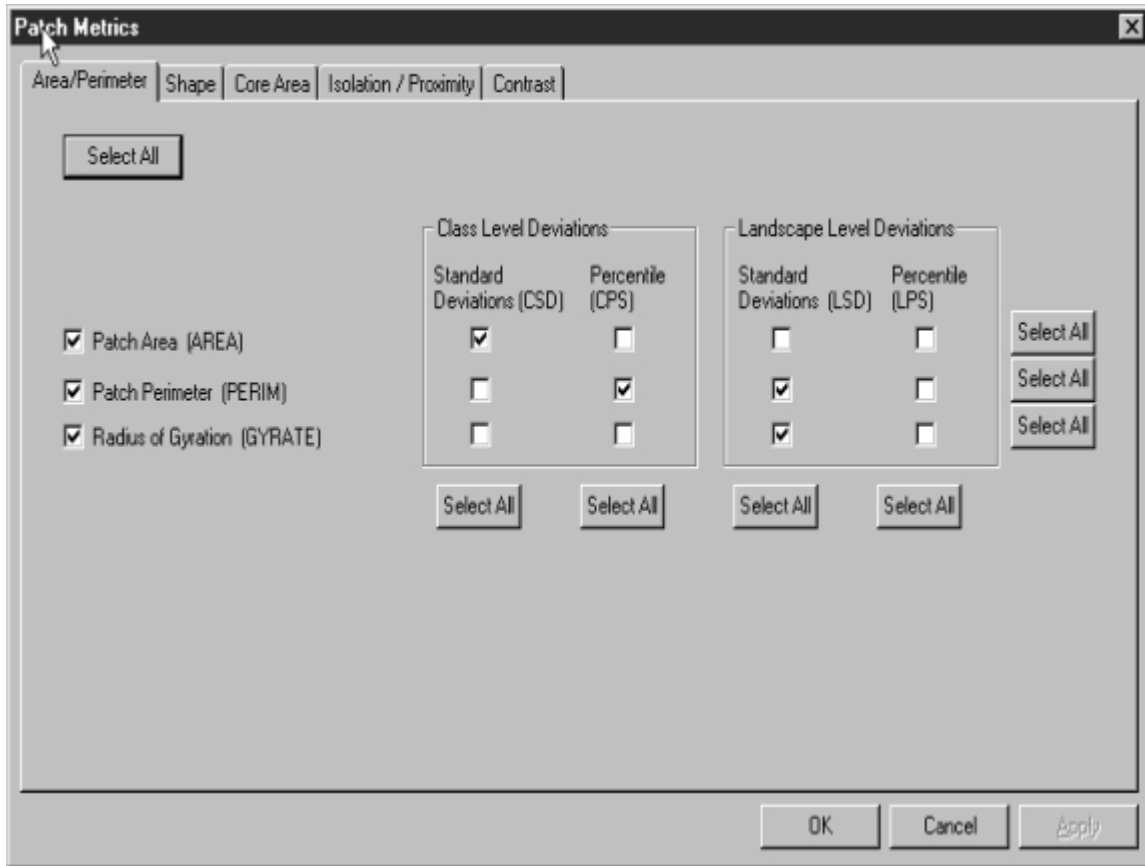


Figure 5. Patch metrics dialog box depicting the Area/Perimeter metrics tab. Each tab corresponds to a different suite of metrics.

The selection of metrics is relatively straightforward. On each tabbed page select the desired individual metrics using the check boxes, or select all of the metrics using the Select All button. If you select all the metrics, the button function will change to a Clear All function. Clicking on the button now will deselect all metrics. Each metric is discussed in detail in the FRAGSTATS Metrics documentation.

A number of metrics require you to provide additional parameters before they can be calculated. In some cases this involves simply selecting among the alternatives provided (e.g., by clicking on a radio button), in other cases this involves entering a number in a text box, and in still other cases, this involves specifying a separate user-supplied file. These special requirements are described below in association with the corresponding tabbed page.

1. *Area/Density/Edge Metrics*.—If you select TOTAL EDGE or EDGE DENSITY on the Area/Density/Edge Metrics tab at either the class or landscape level, you must specify how you want to treat **boundary** and **background** edges (Fig. 6); specifically, what proportion of the landscape boundary and background class edges to treat as *true* edge and therefore included in the edge length calculations? Note, if a border is present, then only background edges are affected by this designation, since all other edges along the landscape boundary will be made explicit by the information in the

border. If a border is absent, then all boundary and background edges are affected. See the Overview section for a more detailed discussion. There are three options:

- (1) **None.**—Do not count any boundary/background as edge.
- (2) **All.**—Count all boundary/background as edge.
- (3) **Partial.**—Specify the proportion of boundary/background edge to treat as an edge (weight = fraction between 0 and 1). For example, if you specify .5, then half the total length of edge of involving the landscape boundary (if a border is absent) and any internal background will be included as edge in the affected metrics.

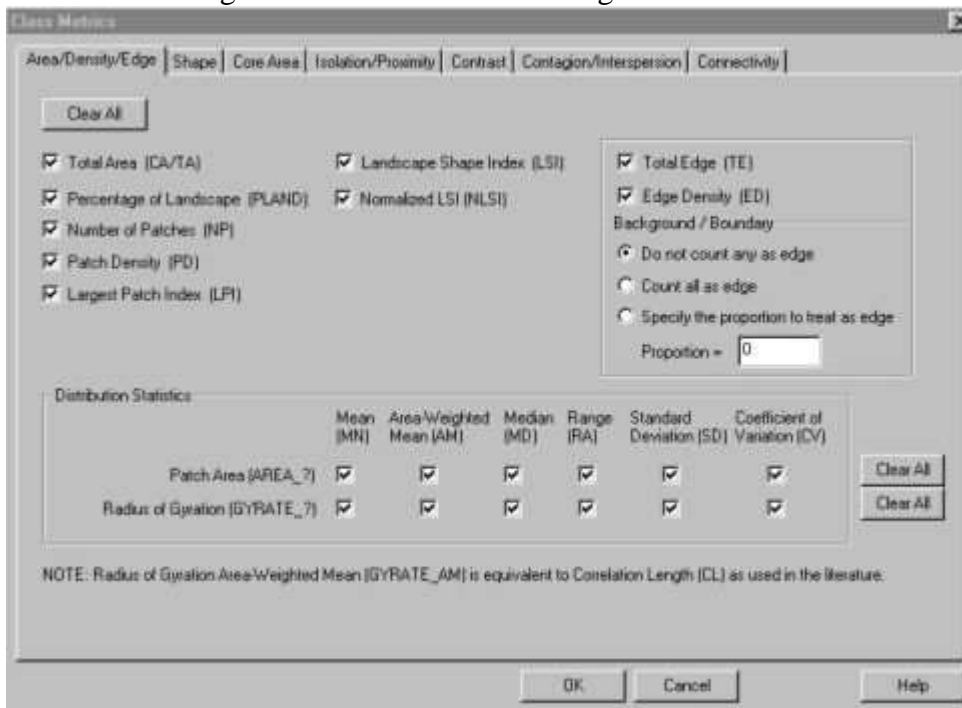


Figure 6.

Class metrics dialog box with the Area/Density/Edge tab showing. The radio buttons associated with background/boundary are active it either Total Edge or Edge Density are checked.

2. **Core Area Metrics.**—If you select any core area metric at the patch, class, or landscape level, you must specify the **depth-of-edge** to use in determining what constitutes the *core* of a patch. There are two options:

- (1) **Enter constant edge depth.**—If you wish to treat all edges the same, then check the corresponding check box and enter a non-zero distance (in meters). By default this box contains a zero, but you must enter a non-zero distance before closing the dialog box or you will receive an error message in the log window. Note, a zero depth-of-edge would result in the core area being equal to patch area, and thus would be redundant.

(2) **Edge depth file.**—Alternatively, you can specify a *comma-delimited ASCII file* that provides unique edge depths for each edge type (i.e., each pairwise combination of patch types). Simply click on the Edge Depth File button and navigate to the appropriate file. The file must be given in the form of a *square two-way matrix*, where each row represents a focal class and each column represents an adjacent or abutting class, and the cell values are the edge depths in meters. The format for this file is as follows:

```
FTABLE, 1stClassID, 2ndClassID, etc.
1stClassID, EdgeDepth_1-1, EdgeDepth_1-2, etc.
2ndClassID, EdgeDepth_2-1, EdgeDepth_2-2, etc.
etc.
```

- FTABLE must be specified in the first cell.
- *1stClassID* is an integer value corresponding to a class value.
- *2ndClassID* is an integer value corresponding to a class value.
- *EdgeDepth<sub>i-j</sub>* is an integer value giving the depth-of-edge (in meters) for the corresponding edge type (i.e., for the focal class designated by the row *ClassID* and the adjacent class designated by the column *ClassID*). Note, it is advisable to provide edge depths in increments equal to the cell size, because FRAGSTATS will always round up or down to the nearest cell when applying the edge mask (see Overview discussion for details).

The edge depth file must be a square matrix (i.e., same number of rows and columns), must have the same list and order of ClassIDs as row headers and column headers, should contain a record for each unique pairwise combination of patch types (classes) in the input landscape (any missing class must be missing in both the rows and columns and will be assigned a zero edge depth for all edges involving that class), and all arguments should be separated by a comma. For example, given four classes the following file would be suitable:

```
FTABLE,1,2,3,4
1,0,30,70,40
2,30,0,40,40
3,30,40,0,50
4,30,40,50,0
```

NOTE, this table can be created and managed using any *spreadsheet* program (e.g., Excel) and then simply saved as a comma delimited file (.csv).

NOTE, this matrix can be *asymmetrical*; that is, upper right and lower left halves do not need to mirror each other. Accordingly, it is important to realize that the rows represent the focal class and the columns represent the adjacent or abutting class. Let's consider the edge depths for focal class 1 in the example above, given in the second row of the table. The edge depths represent the depth-of-edge effect

for patches of class 1 abutting patches of the classes given by the columns. Thus, the class 1 (row)–class 3 (column) edge type has an edge effect of 70 m; that is, the edge effect penetrates 70 m into class 1. Conversely, focal class 3 (row) adjacent to class 1 (column) has an edge effect of 30 m. Thus, the edge effect penetrates less into class 3 than into class 1. This asymmetry may be important in many applications; for example, when urban edge effects penetrate deeply into forest, but forest edge effects penetrate very little, if at all, into urban areas.

NOTE, the *diagonals* are typically given a zero edge depth, but it is possible to specify a nonzero diagonal. However, the only situation in which a patch can abut a patch of the same class is along the landscape boundary when a landscape border is present (see discussion on Background, Boundaries, and Borders). In this case, it is possible to specify a nonzero edge depth, although in most cases it would not be logical to do so.

NOTE, if you have *background* in the image, you need to include the background class value specified in the Run Parameters dialog box in the list of classes specified in the edge depth file, otherwise all background edges will be given a zero edge depth.

- *Isolation/Proximity Metrics*.--If you select either the PROXIMITY INDEX or SIMILARITY INDEX at the patch, class, or landscape level, you must specify the **search radius** (i.e., the distance [in meters] from a focal patch within which neighboring patches are evaluated). By default the search radius is zero, but you must enter a non-zero distance before closing the dialog box or you will receive an error message in the log window. In addition, if you select the similarity index at any level, you must specify a *comma-delimited ASCII file* that provides unique **similarity weights** for each pairwise combination of patch types. Simply click on the Similarity Weight File button to navigate to the appropriate file. Each record in the file should contain a numeric patch type value for each of the two classes and a similarity weight. The syntax for this file is as follows:

FTABLE, *1stClassID*, *2ndClassID*, *etc.*  
*1stClassID*, *SimilarWeight\_1-1*, *SimilarWeight\_1-2*, *etc.*  
*2ndClassID*, *SimilarWeight\_2-1*, *SimilarWeight\_2-2*, *etc.*  
*etc.*

- FTABLE must be specified in the first cell.
- *1stClassID* is an integer value corresponding to a class value.
- *2ndClassID* is an integer value corresponding to a class value.
- *SimilarWeight\_i-j* is a fraction that must range between 0 (no similarity) and 1 (maximum similarity) that gives the similarity weight for the corresponding classes (i.e., for the focal class designated by the row *ClassID* and the adjacent class designated by the column *ClassID*).

The similarity weight file must be a square matrix (i.e., same number of rows and columns), must have the same list and order of ClassIDs as row headers and column headers, should contain a record for each unique pairwise combination of patch types (classes) in the input landscape (any missing class must be missing in both the rows and columns and will be assigned a zero similarity weight for combinations involving that class), and all arguments should be separated by a comma. For example, given four classes the following file would be suitable:

```
FTABLE,1,2,3,4
1,1,0.8,0,0.4
2,0,1,0.6,0.4
3,0.1,0.2,1,0.5
4,0.3,0.4,0.5,1
```

NOTE, this table can be created and managed using any *spreadsheet* program (e.g., Excel) and then simply saved as a comma delimited file (.csv).

NOTE, this matrix can be *asymmetrical*; that is, upper right and lower left halves do not need to mirror each other. Accordingly, it is important to realize that the rows represent the focal class and the columns represent other class combinations. Let's consider the similarity weights for focal class 1 in the example above, given in the second row of the table. The similarity weights represent the degree of similarity between a patch of the focal class and a patch of another class. Thus, a patch of focal class 1 (row) has maximum similarity to a patch of the same class (weight = 1) and a similarity of 0.8 to a patch of class 2, but zero similarity to a patch of class 3. Conversely, a patch of focal class 3 (row) has a similarity of 0.1 to a patch of class 1. In most cases, however, it is more logical to think of similarity in terms of symmetrical weights.

NOTE, the *diagonals* are typically given a weight of one, because the similarity of two patches of the same class is generally assumed to be the maximum, but it is possible to specify a different value.

NOTE, if you have *background* in the image, you need to include the background class value specified in the Run Parameters dialog box in the list of classes specified in the similarity weight file, otherwise all background edges will be given a zero weight.

- *Contrast Metrics*.--If you select any contrast metric at the patch, class, or landscape level, you must specify a *comma-delimited ASCII file* that provides **edge contrast weights** for each pairwise combination of patch types. Simply click on the Edge Weight File button to navigate to the appropriate file. Each record in the file should contain a numeric patch type value for each of the two classes and an edge contrast weight. The syntax for this file is as follows:

```
FTABLE, 1stClassID, 2ndClassID, etc.
```

*1stClassID, ContrastWeight\_1-1, ContrastWeight\_1-2, etc.*  
*2ndClassID, ContrastWeight\_2-1, ContrastWeight\_2-2, etc.*  
*etc.*

- FTABLE must be specified in the first cell.
- *1stClassID* is an integer value corresponding to a class value.
- *2ndClassID* is an integer value corresponding to a class value.
- *ContrastWeight* is a fraction that must range between 0 (no contrast) and 1 (maximum contrast) that gives the edge contrast weight for the corresponding classes (i.e., for the focal class designated by the row *ClassID* and the adjacent class designated by the column *ClassID*).

The edge contrast weight file must be a square matrix (i.e., same number of rows and columns), must have the same list and order of ClassIDs as row headers and column headers, should contain a record for each unique pairwise combination of patch types (classes) in the input landscape (any missing class must be missing in both the rows and columns and will be assigned a zero edge contrast weight for combinations involving that class), and all arguments should be separated by a comma. For example, given four classes the following file would be suitable:

```
FTABLE,1,2,3,4  
1,0,0.8,0.6,0.4  
2,0.8,0,0.7,0.3  
3,0.6,0.7,0,0.5  
4,0.4,0.3,0.5,0
```

NOTE, this table can be created and managed using any *spreadsheet* program (e.g., Excel) and then simply saved as a comma delimited file (.csv).

NOTE, this matrix can be *asymmetrical*; that is, upper right and lower left halves do not need to mirror each other, although in most cases it is more logical to think of edge contrast as being symmetrical. Accordingly, it is important to realize that the rows represent the focal class and the columns represent adjacent or abutting classes. Let's consider the edge contrast weights for focal class 1 in the example above, given in the second row of the table. A patch of focal class 1 (row) adjacent to a patch of class 2 (column) has a contrast weight of 0.8; that is, it is 80% of the maximum contrast (1). Likewise, an edge between focal class 1 (row) and adjacent class 3 (column) has a contrast weight of 0.6 (i.e., 60% of maximum).

NOTE, the *diagonals* are typically given a zero edge contrast weight, but it is possible to specify a nonzero diagonal. However, the only situation in which a patch can abut a patch of the same class is along the landscape boundary when a landscape border is present (see discussion on Background, Boundaries, and Borders). In this case, it is possible to specify a nonzero edge contrast weight, although in most cases it would not be logical to do so.

NOTE, if you have *background* in the image, you need to include the background class value specified in the Run Parameters dialog box in the list of classes specified in the edge contrast weight file, otherwise all background edges will be given a zero weight.

#### **Step 4. Working with Batch Files (optional)**

If you wish to analyze a single landscape, then simply select Landscape mode as the Input File Type in the Run Parameters dialog box (see Step 2) and skip this step. However, if you have many landscapes to analyze, you may want to run FRAGSTATS in batch mode by selecting Batch mode as the Input File Type option in the Run Parameters dialog box (Step 2). If you select Batch mode, you must also specify a properly formatted batch file by clicking on the Batch File Name button and navigating to the appropriate file. Note, FRAGSTATS uses the file extension .fbt for batch files and will look for files with this extension by default when navigating. The .fbt extension is not mandatory, but using it can help keep files organized. The batch file must be a *comma-delimited ASCII file*. Each line should specify the input landscape file name, cell size, background value, number of rows, number of columns, and input data type, in that order. The syntax for this file is as follows:

*InputFileName, CellSize, Background, Rows, Columns, InputDataType*

- *InputFileName* is the full path and file name of the input landscape.
- *CellSize* is an integer value corresponding to the cell size (in meters).
- *Background* is an integer value corresponding to the designated background value. Note, any class designated as background in the Class Properties dialog box (see Step 5 on Modifying Class Properties) will be reclassified to this class value.
- *Rows* is an integer value corresponding to the number of rows in the input image.
- *Columns* is an integer value corresponding to the number of columns in the input image.
- *InputDataType* is a character string identifying the input data format, with the following options corresponding to the various input data format types:
  - IDF\_ARCGRID
  - IDF\_ASCII
  - IDF\_8BIT
  - IDF\_16BIT
  - IDF\_32BIT
  - IDF\_ERDAS
  - IDF\_IDRISI

The batch file should contain a record for each input landscape, and all arguments should be separated by a comma. The parameter values should reflect the Input Data Type for each input landscape. In other words, if the Input Data Type is ASCII or

BINARY, then the record must contain all of the parameters specified above. However, if the Input Data Type is ARCGRID, ERDAS, or IDRISI, then the record need only contain the input landscape file name and the background value; the remaining parameters should be assigned “x”, as illustrated in the following example:

```
D:\Foo\Bar\ASCIIfilename,25,999,250,300,IDF_ASCII
D:\Foo\Bar\8BITfilename2,25,999,250,300,IDF_8BIT
D:\Foo\Bar\16BITfilename2,25,999,250,300,IDF_16BIT
D:\Foo\Bar\32BITfilename2,25,999,250,300,IDF_32BIT
D:\Foo\Bar\ARCGRIDfilename,x,999,x,x,IDF_ARCGRID
D:\Foo\Bar\ERDASfilename,x,999,x,x,IDF_ERDAS
D:\Foo\Bar>IDRISIfilename,x,999,x,x,IDF_IDRISI
etc.
```

NOTE, since Arc Grid files are actually folders containing multiple files, not single image files, the file naming convention is a bit different. In this case, the input file name should be given as the path to the Arc Grid folder. In addition, regardless of input data type, the cell size, background value, rows and columns can be different for each input landscape.

NOTE, running from a batch file does not eliminate the necessity of completing the parameterization of FRAGSTATS; it only provides a mechanism for running FRAGSTATS on more than one landscape without having to parameterize and run each landscape separately. Specifically, you still must set the remaining run parameters (Step 2) and select and parameterize the individual metrics (Step 3). The batch file only specifies the file name, input data type, and grid attributes for each input landscape; all other run parameters must be specified according to the directions in Steps 2 and 3.

### **Using the Batch File Editor**

FRAGSTATS includes a **Batch File Editor** that allows you to build and edit batch files via a dialog box (Fig. 7) accessible via the **Tools** menu in the opening window:

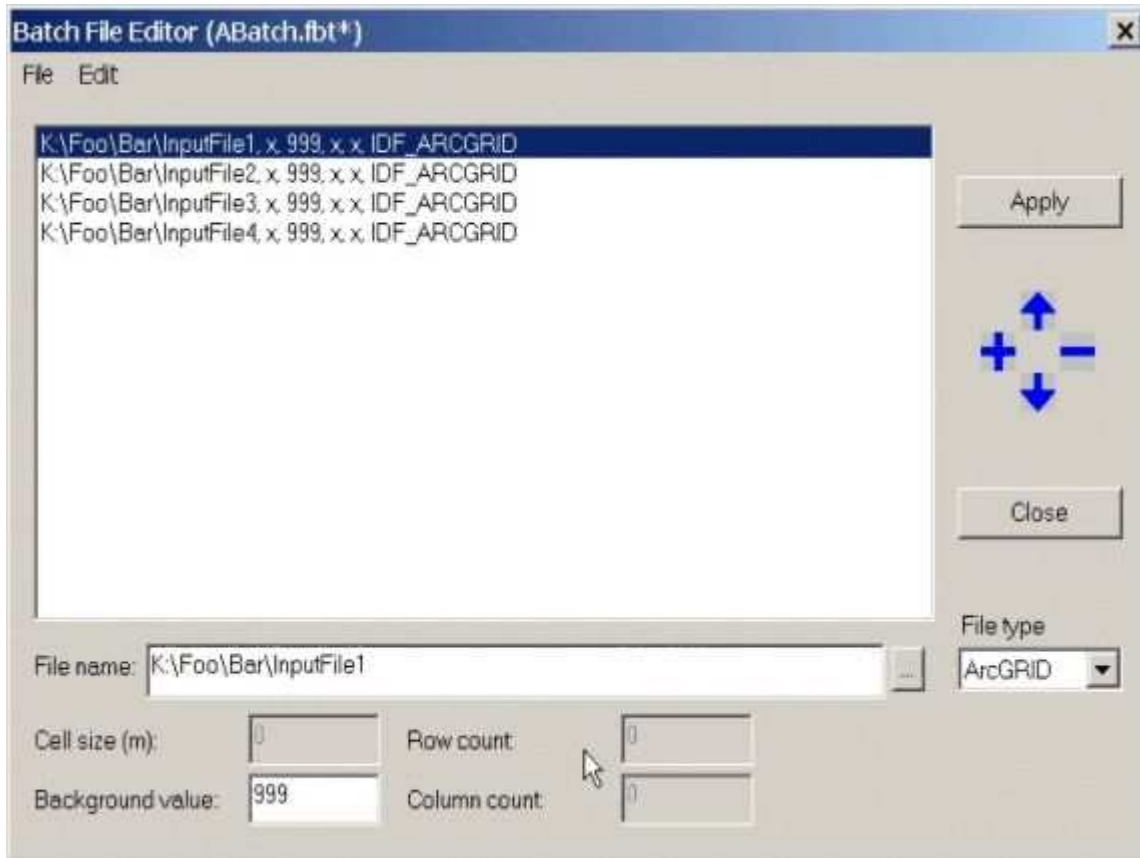


Figure 7. Batch file editor dialog box.

- Using the Batch File Editor to Create a Batch File.—Although in most cases it will be more efficient to create a batch file using a text editor (because you can copy and paste lines easily), you can build one using the Batch File Editor by following these steps:
  1. *Add a Record.*—When you first open the Batch File Editor, the main text window (Fig. 7) will be empty. You must first add a record to the empty batch file by selecting the Add Record option from the **Edit** drop-down menu or by simply clicking on the “+” button to the right of the main text window. When you do so, the text window will display “(invalid entry)” because the rest of the required information has not been entered yet.
  2. *Select File Type.*—Next, select the appropriate input data type from the list of options available by pressing the down arrow button under File Type. For example, if the first input landscape is an Arc Grid, then select the Arc Grid file type.
  3. *Select Input File.*—Next, select an input file by typing in the full path and file name or clicking on the navigation button to navigate to the file. Note, by selecting the File Type first (step 2 above), the navigation will be restricted by default to only files of the specified type.

4. *Enter Relevant Grid Attributes.*—Next, enter the relevant grid attributes given the selected file type. If the File Type is ArcGRID, ERDAS, or IDRISI, then you only need to enter a background value. If the File Type is ASCII or BINARY, then you need to enter the cell size, background value, and number of rows and columns.
  5. *Apply the Specified Parameters.*—Click on the Apply button to the right of the main text window to apply the specified parameters. The record in the text window should change to reflect the specified parameters. Note, if you enter inappropriate values in the text window, the record will be marked as “(invalid entry)” and you will need to make the necessary changes.
  6. *Repeat the Process for each Input Landscape.*—Repeat steps 1-5 for each input landscape that you want to include in the batch file. Note, you can specify different input file types and grid attributes for each record.
  7. *Save and Close Batch File.*—When you are finished adding records to the batch file, and you are certain that all entries are valid, then you can save the batch file. You can save this file and *remain* in the **Batch File Editor** via the **File-Save** or **File-Save As** options, or you can save this file and *close* the Editor in several ways: (1) by clicking on the Close button to the right of the main text window, (2) by clicking on the “X” in upper right corner of the Editor window, or (3) by selecting the **File-Close** option. In all cases, you will be prompted to enter a file name if it is a new batch file, or you will be queried to save the changes if the file name already exists.
- *Using the Batch File Editor to Edit Batch Files.*—In some cases you may need to edit an existing batch file. You can use the **Batch File Editor** to edit an existing file by following these steps:
    1. *Open Existing Batch File.*—Open an existing batch file using the **File-Open** menu item. Note, you can edit a newly created batch file using these same procedures. The main text window (Fig. 7) should display the contents of the batch file. Any invalid records will be labeled as such.
    2. *Modify the Arguments of Existing Records.*—You can modify the arguments associated with each record by changing the relevant parameter and applying the changes. For example, if the selected record does not contain a valid file type argument, select the appropriate file type from the available options by clicking on the down arrow button under File Type, and then click on the Apply button located to the right of the main text window. The file type argument should get updated in the selected record. Any or all of the arguments can be changed before applying the changes.
    3. *Add and Delete Records.*—You can easily add and delete records from the batch file. To add a record simply follow Steps 1-5 described above under ‘Using the Batch File Editor to Create a Batch File’. To delete a record, simply select the

- record and delete it via the **Edit-Delete** menu option or by depressing the “-” button located to the right of the main text window.
4. *Promote and Demote Records.*—You can change the order of records by promoting or demoting their position in the batch file. Simply select the record by clicking on the record in the main text window and then use the **Edit-Promote/Demote** menu options or depress the up or down arrows located to the right of the main text window.
  5. *Save and Close Batch File.*—When you are finished editing the batch file, and you are certain that all entries are valid, you can save the file as before via the **File-Save** or **File-Save As** options. Alternatively, you can save this file and close the Editor in several ways: (1) by clicking on the Close button to the right of the main text window, (2) by clicking on the “X” in upper right corner of the Editor window, or (3) by selecting the **File-Close** option. In all cases, you will be prompted as to whether to save the changes to the existing file.

#### **Step 4. Working with Batch Files (optional)**

If you wish to analyze a single landscape, then simply select Landscape mode as the Input File Type in the Run Parameters dialog box (see Step 2) and skip this step. However, if you have many landscapes to analyze, you may want to run FRAGSTATS in batch mode by selecting Batch mode as the Input File Type option in the Run Parameters dialog box (Step 2). If you select Batch mode, you must also specify a properly formatted batch file by clicking on the Batch File Name button and navigating to the appropriate file. Note, FRAGSTATS uses the file extension .fbt for batch files and will look for files with this extension by default when navigating. The .fbt extension is not mandatory, but using it can help keep files organized. The batch file must be a *comma-delimited ASCII file*. Each line should specify the input landscape file name, cell size, background value, number of rows, number of columns, and input data type, in that order. The syntax for this file is as follows:

*InputFileName, CellSize, Background, Rows, Columns, InputDataType*

- InputFileName* is the full path and file name of the input landscape.
- CellSize* is an integer value corresponding to the cell size (in meters).
- Background* is an integer value corresponding to the designated background value. Note, any class designated as background in the **Class Properties** dialog box (see Step 5 on Modifying Class Properties) will be reclassified to this class value.
- Rows* is an integer value corresponding to the number of rows in the input image.
- Columns* is an integer value corresponding to the number of columns in the input image.
- InputDataType* is a character string identifying the input data format, with the following options corresponding to the various input data format types:

- IDF\_ARCGRID
- IDF\_ASCII
- IDF\_8BIT
- IDF\_16BIT
- IDF\_32BIT
- IDF\_ERDAS
- IDF\_IDRISI

The batch file should contain a record for each input landscape, and all arguments should be separated by a comma. The parameter values should reflect the Input Data Type for each input landscape. In other words, if the Input Data Type is ASCII or BINARY, then the record must contain all of the parameters specified above. However, if the Input Data Type is ARCGRID, ERDAS, or IDRISI, then the record need only contain the input landscape file name and the background value; the remaining parameters should be assigned “x”, as illustrated in the following example:

```
D:\Foo\Bar\ASCIIfilename,25,999,250,300,IDF_ASCII
D:\Foo\Bar\8BITfilename2,25,999,250,300,IDF_8BIT
D:\Foo\Bar\16BITfilename2,25,999,250,300,IDF_16BIT
D:\Foo\Bar\32BITfilename2,25,999,250,300,IDF_32BIT
D:\Foo\Bar\ARCGRIDfilename,x,999,x,x,IDF_ARCGRID
D:\Foo\Bar\ERDASfilename,x,999,x,x,IDF_ERDAS
D:\Foo\Bar\IDRISIfilename,x,999,x,x,IDF_IDRISI
etc.
```

NOTE, since Arc Grid files are actually folders containing multiple files, not single image files, the file naming convention is a bit different. In this case, the input file name should be given as the path to the Arc Grid folder. In addition, regardless of input data type, the cell size, background value, rows and columns can be different for each input landscape.

NOTE, running from a batch file does not eliminate the necessity of completing the parameterization of FRAGSTATS; it only provides a mechanism for running FRAGSTATS on more than one landscape without having to parameterize and run each landscape separately. Specifically, you still must set the remaining run parameters (Step 2) and select and parameterize the individual metrics (Step 3). The batch file only specifies the file name, input data type, and grid attributes for each input landscape; all other run parameters must be specified according to the directions in Steps 2 and 3.

## Using the Batch File Editor

FRAGSTATS includes a Batch File Editor that allows you to build and edit batch files via a dialog box (Fig. 7) accessible via the **Tools** menu in the opening window:

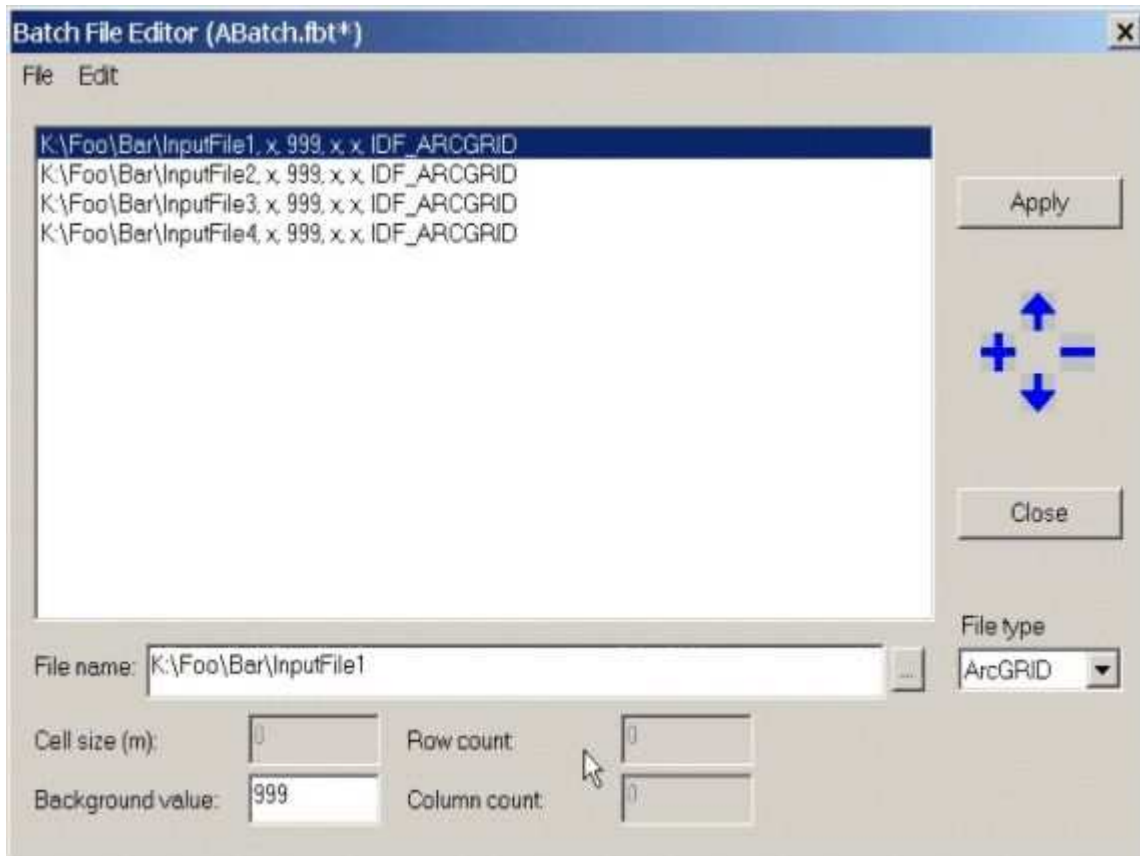


Figure 7. Batch file editor dialog box.

- *Using the Batch File Editor to Create a Batch File.*—Although in most cases it will be more efficient to create a batch file using a text editor (because you can copy and paste lines easily), you can build one using the Batch File Editor by following these steps:
  1. *Add a Record.*—When you first open the Batch File Editor, the main text window (Fig. 7) will be empty. You must first add a record to the empty batch file by selecting the Add Record option from the **Edit** drop-down menu or by simply clicking on the “+” button to the right of the main text window. When you do so, the text window will display “(invalid entry)” because the rest of the required information has not been entered yet.
  2. *Select File Type.*—Next, select the appropriate input data type from the list of options available by pressing the down arrow button under File Type. For example, if the first input landscape is an Arc Grid, then select the Arc Grid file type.

3. *Select Input File.*—Next, select an input file by typing in the full path and file name or clicking on the navigation button to navigate to the file. Note, by selecting the File Type first (step 2 above), the navigation will be restricted by default to only files of the specified type.
  4. *Enter Relevant Grid Attributes.*—Next, enter the relevant grid attributes given the selected file type. If the File Type is ArcGRID, ERDAS, or IDRISI, then you only need to enter a background value. If the File Type is ASCII or BINARY, then you need to enter the cell size, background value, and number of rows and columns.
  5. *Apply the Specified Parameters.*—Click on the Apply button to the right of the main text window to apply the specified parameters. The record in the text window should change to reflect the specified parameters. Note, if you enter inappropriate values in the text window, the record will be marked as “(invalid entry)” and you will need to make the necessary changes.
  6. *Repeat the Process for each Input Landscape.*—Repeat steps 1-5 for each input landscape that you want to include in the batch file. Note, you can specify different input file types and grid attributes for each record.
  7. *Save and Close Batch File.*—When you are finished adding records to the batch file, and you are certain that all entries are valid, then you can save the batch file. You can save this file and *remain* in the Batch File Editor via the **File-Save** or **File-Save As** options, or you can save this file and *close* the Editor in several ways: (1) by clicking on the Close button to the right of the main text window, (2) by clicking on the “X” in upper right corner of the Editor window, or (3) by selecting the **File-Close** option. In all cases, you will be prompted to enter a file name if it is a new batch file, or you will be queried to save the changes if the file name already exists.
- *Using the Batch File Editor to Edit Batch Files.*—In some cases you may need to edit an existing batch file. You can use the Batch File Editor to edit an existing file by following these steps:
    1. *Open Existing Batch File.*—Open an existing batch file using the **File-Open** menu item. Note, you can edit a newly created batch file using these same procedures. The main text window (Fig. 7) should display the contents of the batch file. Any invalid records will be labeled as such.
    2. *Modify the Arguments of Existing Records.*—You can modify the arguments associated with each record by changing the relevant parameter and applying the changes. For example, if the selected record does not contain a valid file type argument, select the appropriate file type from the available options by clicking on the down arrow button under File Type, and then click on the Apply button located to the right of the main text window. The file type argument should get

updated in the selected record. Any or all of the arguments can be changed before applying the changes.

3. *Add and Delete Records.*—You can easily add and delete records from the batch file. To add a record simply follow Steps 1-5 described above under ‘Using the Batch File Editor to Create a Batch File’. To delete a record, simply select the record and delete it via the **Edit-Delete** menu option or by depressing the “-” button located to the right of the main text window.
4. *Promote and Demote Records.*—You can change the order of records by promoting or demoting their position in the batch file. Simply select the record by clicking on the record in the main text window and then use the **Edit-Promote/Demote** menu options or depress the up or down arrows located to the right of the main text window.
5. *Save and Close Batch File.*—When you are finished editing the batch file, and you are certain that all entries are valid, you can save the file as before via the **File-Save** or **File-Save As** options. Alternatively, you can save this file and close the Editor in several ways: (1) by clicking on the Close button to the right of the main text window, (2) by clicking on the “X” in upper right corner of the Editor window, or (3) by selecting the **File-Close** option. In all cases, you will be prompted as to whether to save the changes to the existing file.
- 6.

## **Step 6. Executing FRAGSTATS**

Now you are finally ready to execute FRAGSTATS. By selecting the Execute option in the **Fragstats** drop-down menu on the opening window, or clicking on the Execute button on the Tool bar, the program is off and running. An “execution finished” message appears when the processing is complete. Note, a progress indicator on the Status bar displays the progress of the run. For a single landscape Input File Type, a green progress bar is displayed. For a batch Input File Type, a message is displayed that indicates the record number (i.e., input landscape) being processed.

## **Step 7. Browsing and Saving the Results**

The final step is to browse the output and, if desired, save it to a file (actually several files). The **Browse** dialog box allows you to quickly view and evaluate the results of an analysis before actually saving it to a file(s). This can save you the time of opening the results in a separate text editor or importing the data to a spreadsheet before “seeing” the output. Note, you can automatically save results to output files by checking the automatically save results check box in the Run Parameters dialog box and entering a “basename” for the output files. The **Browse** tool (Fig. 9) is accessed via the **Tools** menu in the opening window.

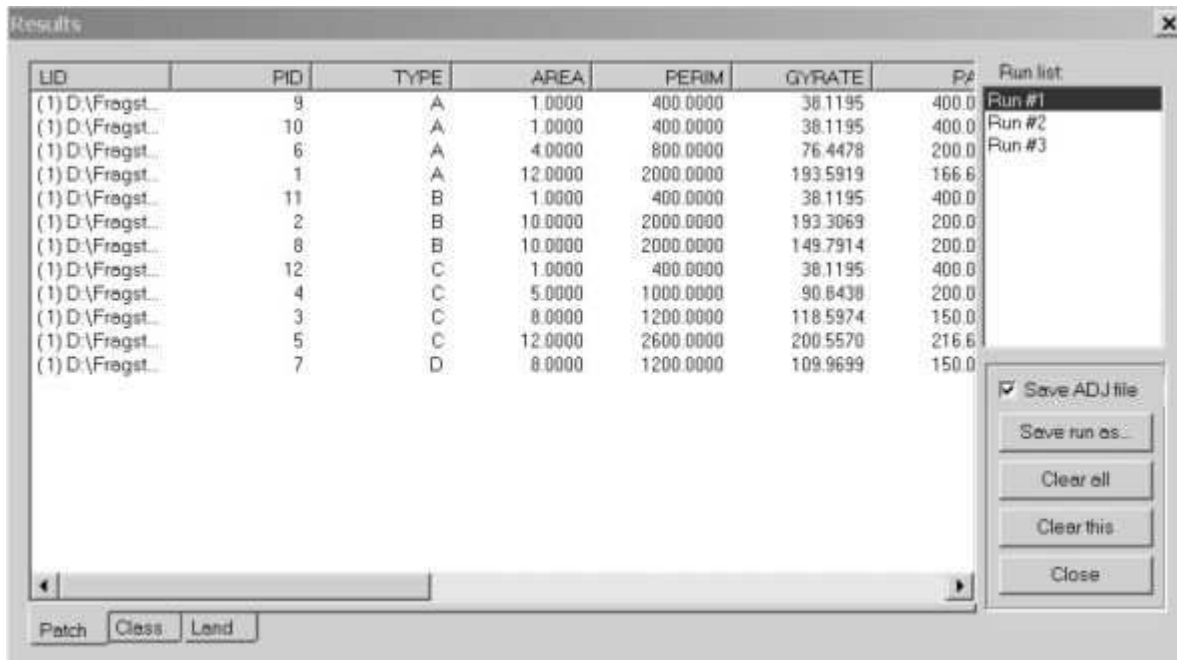


Figure 9. Browse dialog box.

The **Browse** dialog box consists of three main parts: (1) run list, (2) tabbed window displaying patch, class, and landscape metrics, and (3) options for deleting and saving results.

1. Run list.—The Run list window lists the FRAGSTATS runs completed using the currently open parameterization file. Each time you execute FRAGSTATS (step 6), the results are associated with a run number. The first time you execute after parameterizing FRAGSTATS (steps 2-5), the results will be associated with Run 1. The next time you execute, e.g., after modifying one or more run parameters (step 2) or selecting a different suite of metrics (step 3), the results will be associated with Run 2. And so on for every subsequent run. To view the results of a particular run, simply click on the run number in the run list and the results displayed in the metrics display window will change accordingly. If you open a new parameterization file (**File-New**) or open a different parameterization file (**File-Open**), the **Browse** dialog box will be cleared of all runs and the associated data.
2. Metrics Display Window.—The metrics display window contains three tabs for displaying the three levels of metrics: patch, class, and landscape. Click on the corresponding tab to change the level displayed. Note, the tabs will only contain data if the corresponding metrics were selected for the analysis.
3. Deleting and Saving Results.—The results of a run can be cleared from the display or saved to file. The following options are available:

(1) **Save Run As.**—Saves the associated output files to a location and “basename” of your choice. Note, the “basename” will be combined with the .patch, .class, and .land extensions for the corresponding output files, as well as the .adj extension for the adjacency matrix if the save ADJ file box is checked. If you had entered an output file name in the Run Parameters dialog box (step 2), then it will be used as the default basename here; although you can change it here if desired. If you attempt to save to a file name that already exists, you will be prompted as to whether you want to overwrite this file or not. Note, saving to a file of the same name will not append results to the existing file (as in FRAGSTATS 2), because there is no guarantee that the output file structure will be the same. If you wish to append the results of several runs with identical output formats, then you should build a batch file and run in Batch mode. The results from a batch file will be output to a single file because the structure is guaranteed to be the same for every input landscape.

(2) **Clear All.**—Clears all the runs from the **Browse** window. Note, once deleted, these runs cannot be recovered.

(3) **Clear This.**—Clears only the currently selected run (in the run list) from the **Browse** window. Again, once deleted, this run cannot be recovered.

(4) **Close.**—Closes the **Browse** window. Closing the **Browse** window does not delete the runs or the associated data, it simply closes the window so that you can conduct additional runs. Reopening the **Browse** window will display these runs again. Note, however, that if you close the **Browse** window and then open a new parameterization file (**File-New**) or open a different parameterization file (**File-Open**), the **Browse** dialog box will be cleared of all runs and the associated data.