

**Last Updated:** 23 January 2015

**Prepared by:** Kevin McGarigal

## **FRAGSTATS 4 Tutorial**

This tutorial is intended to provide FRAGSTATS users a "quick start" on how to use the software. All of the necessary data and files for the tutorial have been provided and these can be used as templates for how to format your own files latter on. However, this tutorial is not a substitute for the complete FRAGSTATS user manual; all serious FRAGSTATS users are responsible for understanding all of the information in the user manual.

The tutorial is actually a series of short tutorials designed to demonstrate some of the basic features of FRAGSTATS; it is not intended to be a comprehensive guide, but rather a guide to help new users get started. The tutorials include the following:

1. **Setting Up Software and Inspecting Grids** -- Covers the basic setup of the software and computer for running FRAGSTATS and an inspection of the grids that will be used in the subsequent tutorials. All users should complete this tutorial.
2. **Analyzing a single grid** -- Covers the essential steps involved in analyzing a single grid, including the use of ancillary tables for calculating functional metrics.
3. **Batch processing multiple grids** -- Covers the use of batch files for processing multiple grids at once.
4. **Moving window analysis** -- Covers the use of a moving window analysis to create local landscape structure gradients.
5. **Running the command line version from R** -- Covers the execution of FRAGSTATS command line version from R (programming environment and statistical computing environment).
6. **Using sampling strategies to analyze sub-landscapes** -- Covers the use of various sampling strategies to analyze sub-landscapes, including 1) exhaustive sampling using a user-provided, moving window, or systematic tiling scheme, and 2) partial sampling based on user-specified windows around user-provided or random sample points.

# Tutorial 1. Setting Up Software and Inspecting Grids

In this tutorial, you will setup the software and inspect the grids to be analyzed in the subsequent tutorials.

## 1. Download and install FRAGSTATS

First, if you haven't already done so, download FRAGSTATS 4.x and run the setup utility to install the software on your computer.

## 2. [Optional] Setup your Computer for use with ESRI ArcGIS

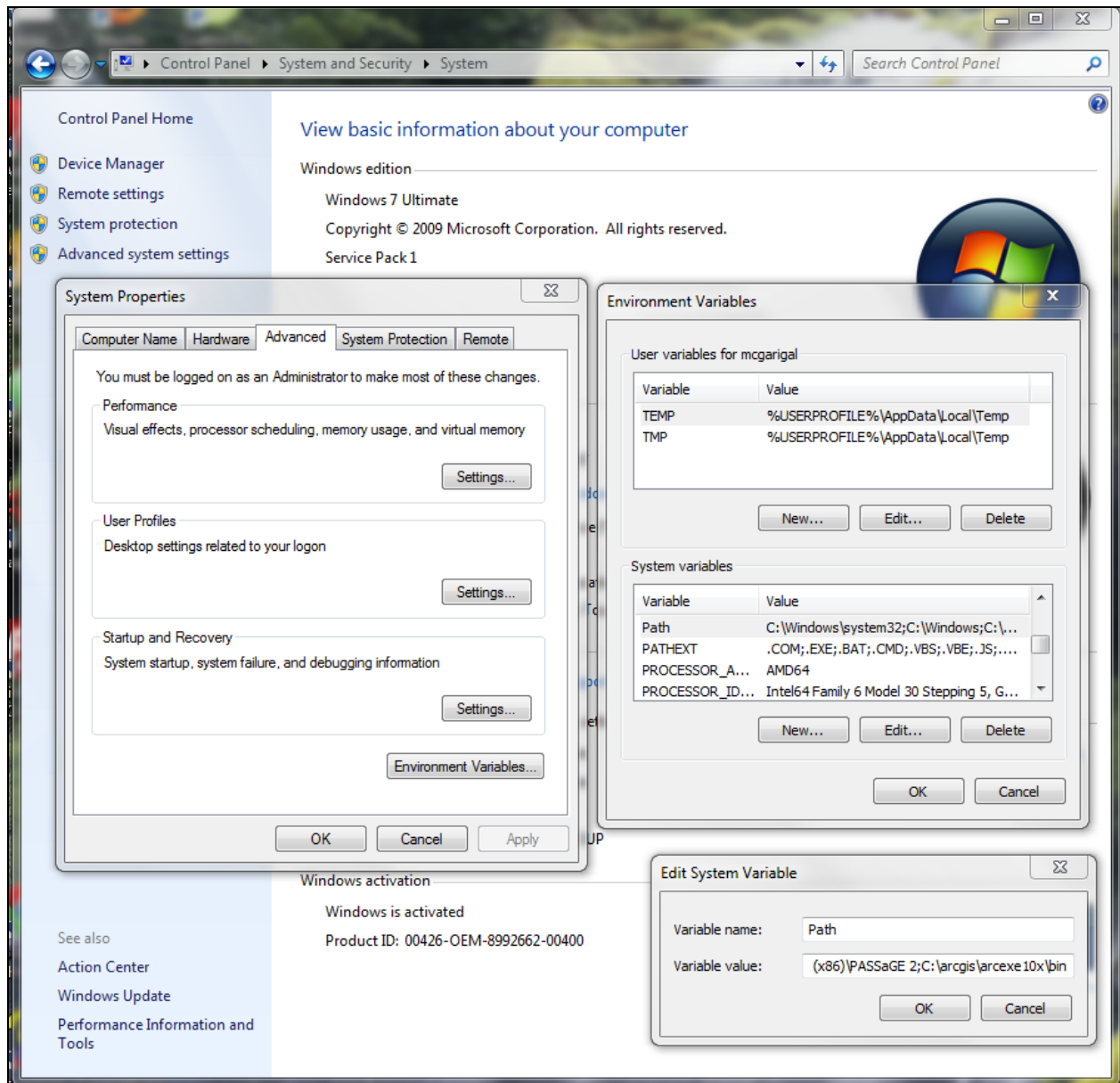
If you intend to work with geotiff (preferred format), ascii, or any of the other data formats except ESRI ArcGIS data, you can skip this step. However, if you have a valid ESRI ArcGIS license (**version 10 or earlier**) with Spatial Analyst or ArcView 3.3 Spatial Analyst and intend to work with ArcGrids (or Rasters), then there are two important requirements, as follows:

**First**, you need to edit your computer system's environmental "path" variable. Specifically, FRAGSTATS must have access to the *aigridio.dll* library found in the "bin" (for ArcGIS installation) or the *avgridio.dll* library found in the "bin32" (for ArcView 3.3 installation) directory. Note, the paths may differ depending on your version and installation. Search your computer for the corresponding file and copy the path to the bin or bin32 directory, as appropriate. Note, the path does NOT include the *aigridio.dll* or *avgridio.dll* file name; it ends with bin or bin32. For example, for an ArcGIS 10 installation, the path might look like: C:\Program Files (x86)\ArcGIS\Desktop10.0\Bin

The path to the corresponding bin directory should be specified in the windows system environmental variable, as follows:

In Windows 7, the Environment variables can be accessed and edited from the Control Panel - System and Security - System - Advanced system settings under the "Advanced" tab and by clicking on "Environment Variables" (at the bottom of the dialog on the "Advanced" tab), as show in the figure below. In the list of System variables, select the "path" variable and select "edit" and add the path to the corresponding bin directory (e.g., ; C:\Program Files (x86)\ArcGIS\Desktop10.0\Bin). Note, you need a semicolon between each path in the list and make sure you enter the correct path on your system.

If you are using a different Windows operating system, you'll have to figure out how to find the system environment variables, then edit the *path* variable as above.



**Second**, if you intend to use the ArcGrids with ArcGIS version 10.0 or earlier you need to make sure that the ArcGrids included with this tutorial are located in a directory on your disk that does NOT contain any spaces in the full path. For example, if you have the tutorial grid reg78b located in the following directory:

C:\Documents and Settings\Fragstats\Tutorial\_1\reg78b

you will get the following error message when attempting to Run FRAGSTATS:

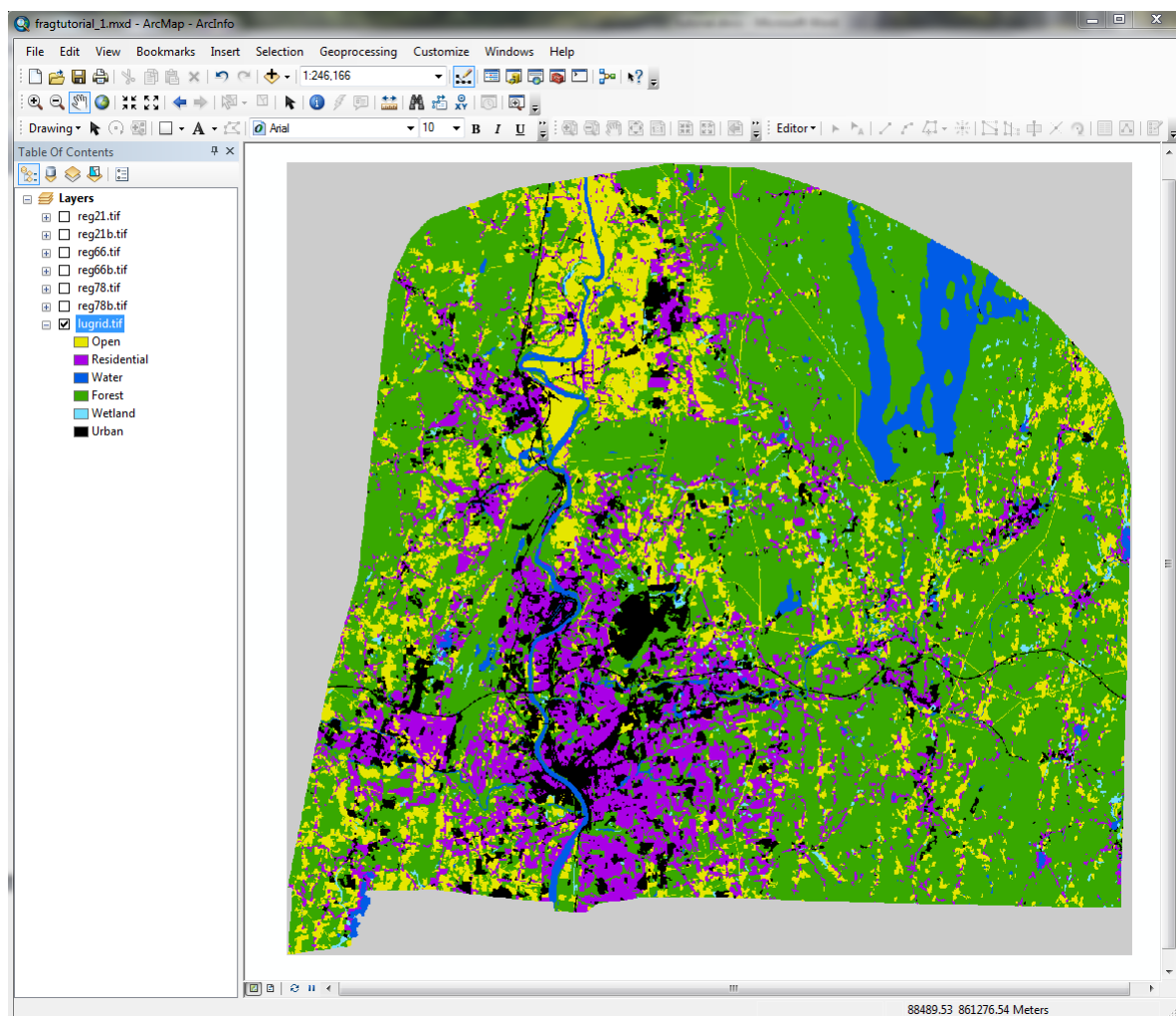
Error: Unexpected error encountered: [cannot\_set\_access\_window for: C:\Documents and Settings\Fragstats\Tutorial\_1\reg78b]. Model execution halted.

This is an irreconcilable problem with ESRI and the only solution is to put all your grids, including those included in this tutorial, in a directory that does NOT contain spaces. Note, this is probably good practice anyways, since there are other software programs that cannot deal properly with spaces in the path.

### 3. Inspect the grids

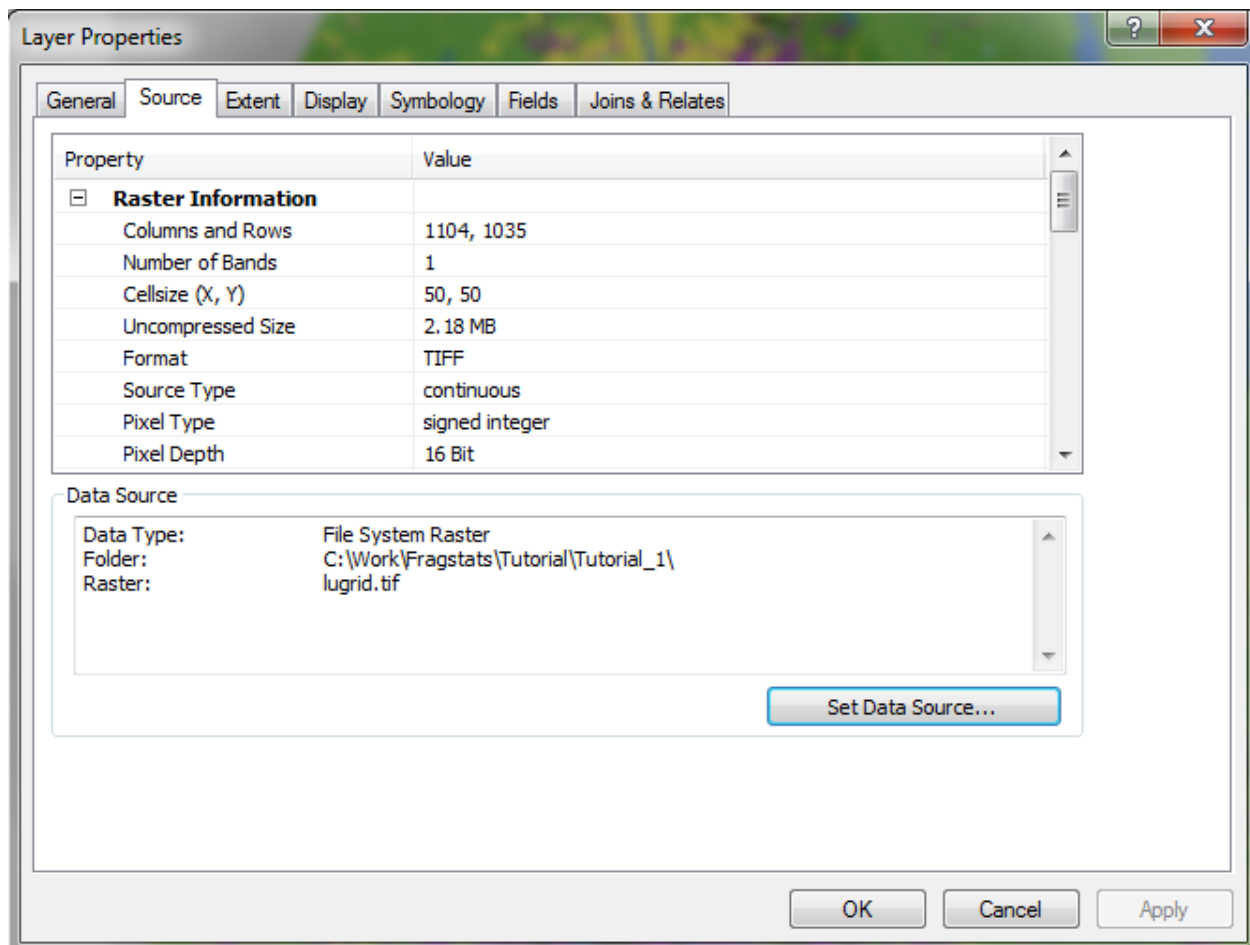
Next, inspect the grids to make sure you understand the landscape definition before analyzing them, since the results of the analysis can only be interpreted in the context of the landscape definition.

**Geotiff/ArcGIS grids.**--If you are planning on working with Geotiffs (preferred format) or ESRI ArcGIS grids and have ESRI ArcGIS installed, open up the provided **fragtutorial\_1.mxd** project in ArcMap. Note, if you are using ArcGIS 9.3, open the provided **fragtutorial\_1\_9.3.mxd**. The project contains several data layers, as listed in the table of contents, including a landcover grid (**lugrid.tif**) for an arbitrary extent in western Massachusetts, and all are Geotiffs.



As you can see from the legend, the **lugrid.tif** contains six landcover classes, including: 1) open (largely agriculture), 2) residential, 3) water (open water bodies and large rivers), 4) forest, 5) wetland, and 6) urban.

Open the lugrid layer properties and inspect the grid properties on the Source tab. In particular, note the grid dimensions (1104 columns by 1035 rows), cell size (50 m), format (GRID), and pixel type (signed integer). The *signed* integer pixel type is necessary if the landscape has a border; i.e., strip of classified cells outside the landscape boundary and assigned negative class values. If the landscape does not contain a border, then an *unsigned* integer type is OK. In this case, the lugrid does not contain a border, but the sub-landscapes (below) do, so the pixel type has been set to signed integer.



Next, open the lugrid layer attribute table and inspect the class values present on the grid. Note the class values and text description for each class. You will need to know the class values later on.

Next, view the **reg78b.tif** grid by selecting it in the table of contexts and zooming to the layer extent. This grid is a randomly selected roughly 5x5 km (25 km<sup>2</sup>) square sub-landscape sampled from the lugrid. Expand the legend and note the landcover classes present; it has the same six landcover classes as before, but with the addition of six "border" classes.

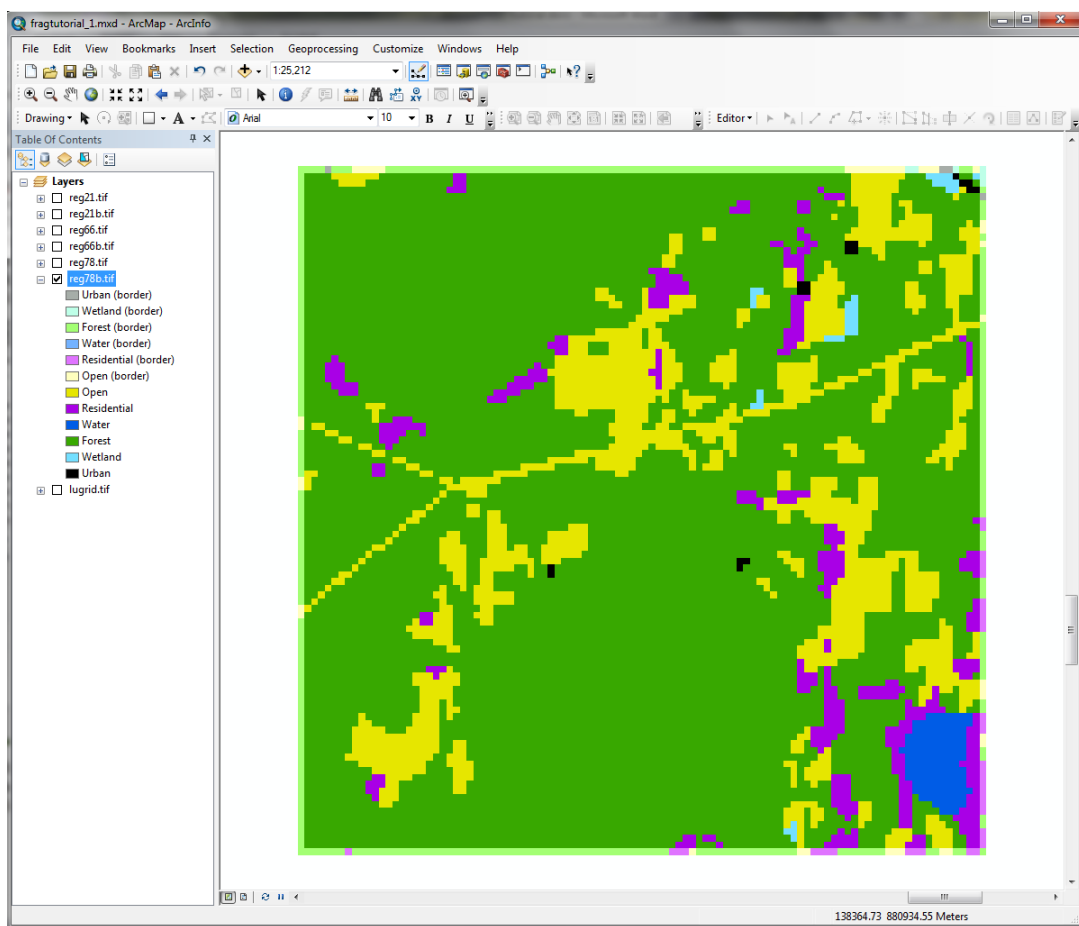
Table

lugrid

Rowid	VALUE *	COUNT	TEXT
0	100	131289	open
1	300	128544	residential
2	400	44128	water
3	500	531043	forest
4	600	9585	non-forested wetland
5	700	63394	urban

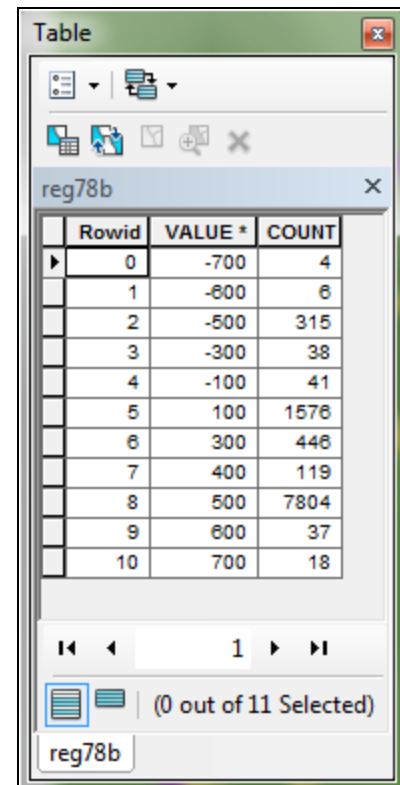
1 (0 out of 6 Selected)

lugrid



The **border** is simply a strip of classified cells surrounding the landscape of interest that provides information on patch type adjacency along the landscape boundary. In the legend provided, the border classes have been assigned a lighter shade of the color assigned to the corresponding class inside the landscape boundary. Importantly, a border is identified in FRAGSTAST by negatively valued cells. An inspection of the grid

attribute table reveals the same six landcover classes as before, but with both a positive (inside the landscape boundary) and negative (border, or outside the landscape boundary) version of each class. Note, I created all three of the grids in this project containing borders (reg78b, reg66b and reg21b) using the **fragborder.aml** script provided. This aml (arc macro language) is executed from the arc prompt and requires ArcInfo Workstation. Briefly, the script clips the lugrid layer with a polygon coverage for one of the sub-landscapes, and then buffers the polygon by 50 m and clips the lugrid layer again but with the buffered (i.e., slightly larger) polygon. Next, the larger grid is multiplied by -1 to convert the cell values to negative. Lastly, the two grids are merged, with the smaller grid (the sub-landscape of interest) on top, resulting in positive values everywhere except the narrow strip of cells in the border.



Rowid	VALUE *	COUNT
0	-700	4
1	-600	6
2	-500	315
3	-300	38
4	-100	41
5	100	1576
6	300	446
7	400	119
8	500	7804
9	600	37
10	700	18

Lastly, view the other sub-landscape grids in the table of contents. There are three different sub-landscapes: reg78, reg66, and reg21, each of which also contains a version with a landscape border: reg78b, reg66b, and reg21b. These landscape differ largely in the amount of forest landcover.

**Ascii/binary grids.**--If you are planning on working with ascii or binary grids, open up the provided files in a text editor and inspect them. Ascii files are interpretable, binary files are not. Neither are pretty to look at and you can't do much with them, but it is useful to know what these files look like.

Open up **reg78b2.asc** in a text editor (the top left portion of this file is shown below). This a space-delimited ascii file (i.e., there is a space between each cell entry) and is therefore interpretable. Note, this ascii file was created by converting the reg78b ArcGrid to an ascii file in ArcMap. Note the header information included in the first six lines of the file. This header information must be deleted before it can be analyzed in FRAGSTATS; however, the information on the grid dimensions (ncols=102 and nrows=102), cellsize (50), and nodata value (-9999) will be needed later when parameterizing the FRAGSTATS model. In particular, note the landscape border indicated by the negative class values in the first row and column.



```

ncols      102
nrows      102
xllcorner  137882.625
yllcorner  875599.5625
cellsize   50
NODATA_value -9999
-500 -500 -500 -500 -700 -500 -500 -500 -100 -100 -100 -100 -100 -500 -500 -500 -500 -500 -500 -
-500 500 500 500 500 100 100 100 100 100 100 100 500 500 500 500 500 500 500 500 300
-500 500 500 500 500 500 100 100 100 100 500 500 500 500 500 500 500 500 500 500 300
-500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 300
-500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500
-500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500
-500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500
-500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500
-500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500
-500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500
-500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500
-500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500
-500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500 500

```

Viewing the ascii grids is a bit more difficult without importing them into your favorite GIS. However, if you are an R user, you can use the following script (or open the provided script, **tutorial\_1.R**) to plot the grid in R. Note, there are several ways to plot the grids in R. If you are familiar with the Raster package, you can import the ascii grids and plot them quite easily, but specifying a color scheme for the legend and plotting a pretty legend is a bit tricky. The following script makes use of the base functions in the Graphics library:

First, set the working directory to wherever you have installed the tutorial; e.g.:

```
setwd('c:/work/fragstats/tutorial/tutorial_1')
```

Next, read in the ascii grid, as a matrix, into an object (m):

```
m<-as.matrix(read.table('reg78b.asc'))
```

Next, in order to assign colors to each landcover class, identify each unique class value:

```
uv<-sort(unique(as.vector(m)))
```

Next, create breaks for assigning colors to class values (breaks are at the minimum -1, midpoints and maximum +1). Note, this is necessary because the plot function (image) is designed for continuous variables not categorical variables as is the case with the landcover image:

```
my.breaks<- (c(min(uv)-2, uv) + c(uv, max(uv)+2 ))/2
```

Next, create a color legend for the plot:



```
my.colors<-c('gray','lightskyblue','lightgreen','lightpink','lightyellow','yellow',  
            'purple','slateblue','green','skyblue','black')
```

Next, check to make sure you have a color for every unique class value:

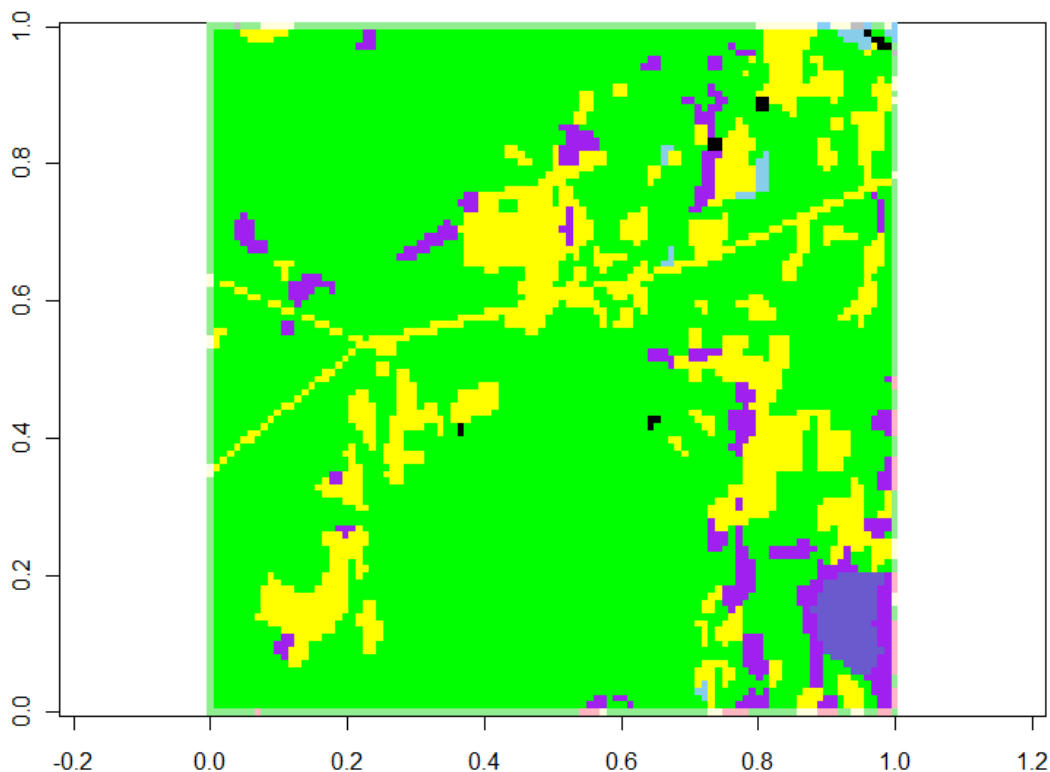
```
if(length(my.colors) != length(uv)) stop("You need a color for every unique  
value")
```

Next, print to the console the color associated with each class value to verify that you have what you want:

```
data.frame(code=uv, color=my.colors)
```

Finally, plot the image with the `image()` function in the `graphic` library. Note, because the `image()` function does a 90 degree counter-clockwise rotation of the image, a matrix transpose and some indexing is necessary to rotate the image back to its original orientation:

```
image(t(m)[,nrow(m):1],asp=1,breaks=my.breaks,col=my.colors)
```

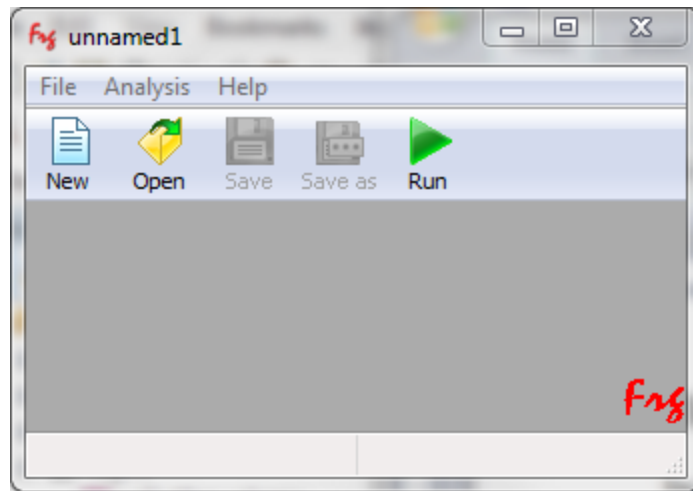


## Tutorial 2. Analyzing a Single Grid

In this tutorial, you will compute a suite of structural and functional patch, class and landscape metrics for a single input grid.

### 1. Open FRAGSTATS

First, open FRAGSTATS from the start menu or by double clicking on the FRAGSTATS icon on the desktop. Note, if you have an ESRI ArcGIS (**v10.0 or earlier**) installation, there may be a significant delay (up to 30 seconds in some cases) in opening the user interface as FRAGSTATS tries to secure a valid ArcGIS license, so be patient -- and don't forget to thank ESRI for their license management while you're waiting.



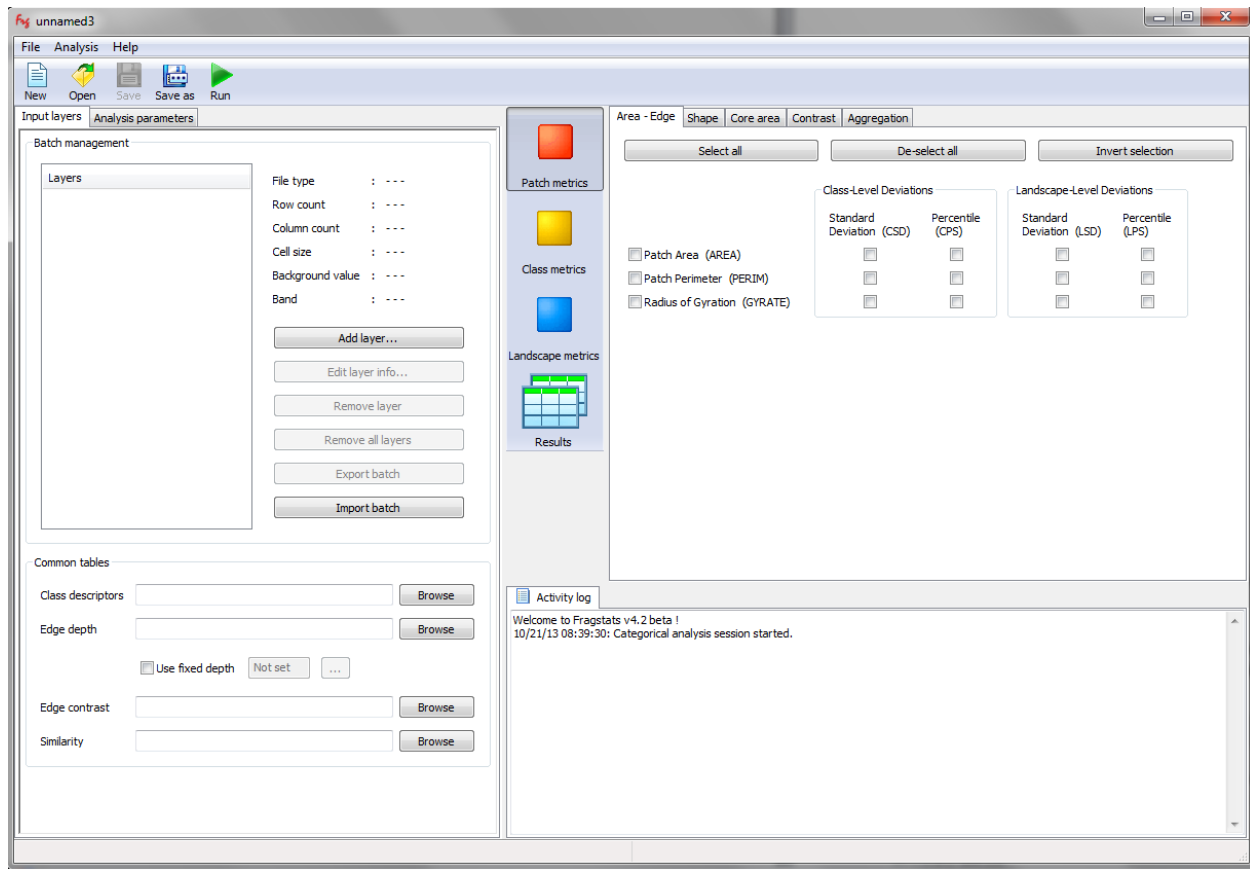
If FRAGSTATS does not open or you get a runtime initialization error or any other kind of error, it probably means that you have ArcGIS installed on your machine and there is problem securing a valid license for Spatial Analyst. If you are sure that you followed the installation instructions and/or followed the instructions in Tutorial #1 step 2, then download the **Diagnostic Tool** from the FRAGSTATS website and after extracting the utility from the zip file, run the utility and copy the report and submit it in an email to [mcgarigalk@eco.umass.edu](mailto:mcgarigalk@eco.umass.edu), along with a detailed description of your setup.

Note, if you have ArcGIS installed on your machine, but either you don't have a valid Spatial Analyst license or you simply don't want to run FRAGSTATS with ArcGrids, then be sure that your system's environmental "path" variable does **NOT** contain a path to ArcGIS. If your system's path variable contains ArcGIS, FRAGSTATS thinks you are an ArcGIS user and that you want to use ArcGrids, and it tries to secure a valid Spatial Analyst license. Unfortunately, to use FRAGSTATS in this situation, you will have to remove ArcGIS from your system's path variable.

### 2. Create a New model

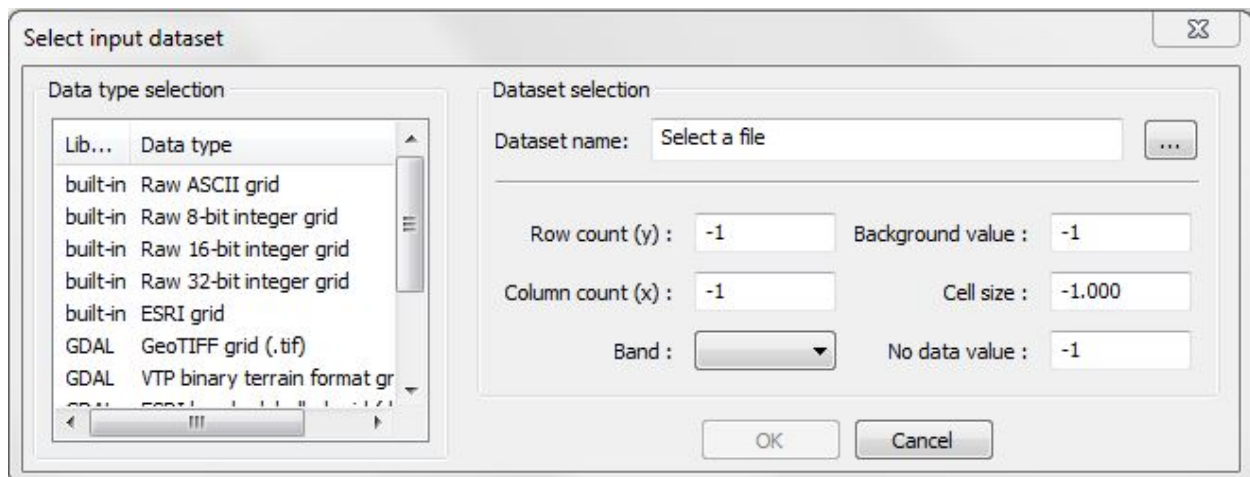
Next, you need to create a FRAGSTATS "model" for a categorical landscape representing the patch mosaic model of landscape structure. A FRAGSTATS model is simply a file containing a complete parameterization of FRAGSTATS; i.e., everything needed to

conduct an analysis. Click on the **New** button on the tool bar or select **New** from the File drop-down menu. This creates a blank model for you to parameterize.

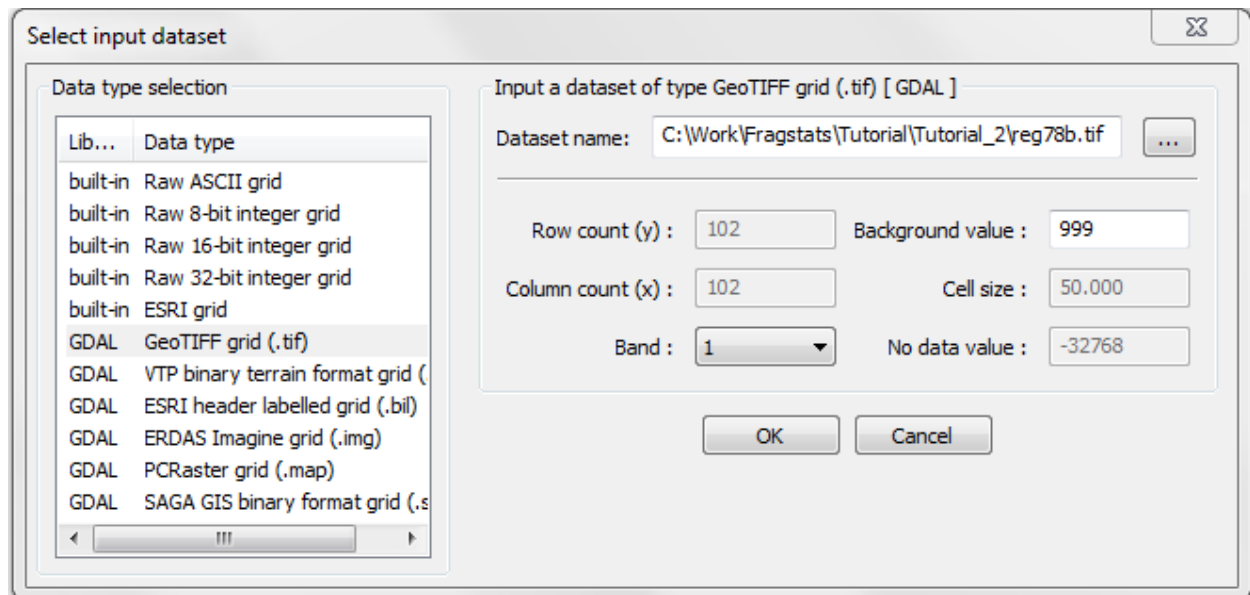


### 3. Import a grid

Next, import a grid to analyze. Specifically, you need to Add a layer to the batch manager on the Input layers tab. Click on the **Add layer** button to open the import data dialog.

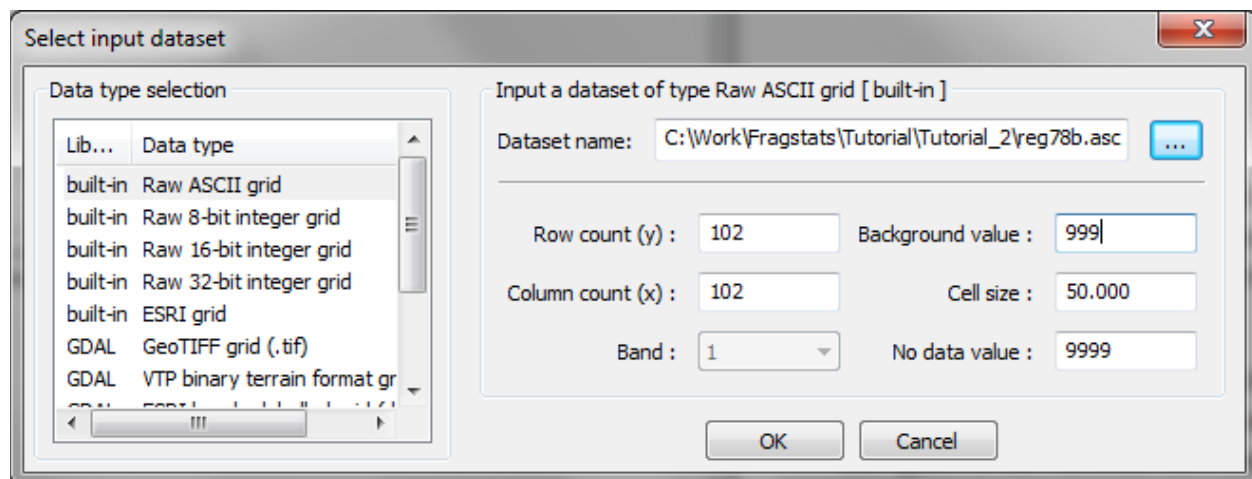


Select **GeoTIFF** data type in the left pane and then navigate to the tutorial directory by clicking on the [...] button and selecting the **reg78b.tif** grid. Note, when you load a Geotiff (or ArcGrid), the grid attribute information pertaining to row count (y), column count (x), cell size, and nodata value are read from the grid header itself, and thus these fields are grayed out in the dialog. The only grid attribute item that you need worry about (and can modify) is the background value.



By default the background class value is set to 999, but you can change it here to any class value that you want, so long as you understand the implications. Briefly, background is a class used to distinguish cells that you essentially want to ignore in the analysis; these can be cells that couldn't be classified to a real landcover class for lack of data, or cells that you simply want to treat as part of the background matrix in the landscape. Importantly, background cells can be considered 'internal' or 'inside' the landscape of interest (if assigned positive values) and/or 'external' or 'outside' the landscape of interest (if assigned negative values). Internal background is considered part of the landscape of interest and contributes to the total landscape area, and thus affects many metrics; external background is not considered to be part of the landscape of interest and only contributes to edge adjacency information for cells along the landscape boundary. To fully understand the implications of designating background, be sure to read the help files on nodata, backgrounds, borders, and boundaries in the section on User guidelines - Overview. Importantly, as a general rule, you should never set the background value equal to the nodata value. If you set the background value equal to the nodata value, and you have internal background, FRAGSTATS cannot distinguish between them and all background (internal and external) and nodata will be treated the same, as external background. For now, keep the background value set to **999**.

If you are using **ascii** or **binary** files, you can select the corresponding data type in the left pane and then navigate to the tutorial directory by clicking on the [...] button and selecting the corresponding grid. For example, to use the provided ascii grid, select **reg78b.asc**. Note, when you try to load an ascii grid or binary grid, the grid attribute information must be entered manually. If you don't enter this information before selecting the grid, the software will complain that the layer attributes are invalid, so be sure to enter valid numbers for each of the attributes after selecting the grid. Specifically, for this grid, you need to enter row count (y) = **102**, column count (x) = **102**, cell size = **50**, and nodata value = **9999**. As with ArcGrids, you can also edit the background class value, but for now, keep the default value of **999**.



#### 4. Optionally, input a class descriptors table

Next, you have the option of inputting a class descriptors table. The class descriptors table allows you to specify a character description (i.e., patch type) for each numeric class value, specify whether to compute statistics for each class, and whether to designate each class as background. The class descriptors table is optional. If you do not provide this table, then the numeric class values are used in the output, all classes are enabled and none are treated as background except any class with the assigned background value (999 in this case).

Open up the provided **descriptors.fcd** file in a text editor. Note, FRAGSTATS recognizes .fcd as the extension for class descriptor files, but this is not a required extension. This file contains four fields. *ID* refers to the numeric class values; these are the unique cell values in the grid. These values derive from your landscape definition. *Name* is simply a description of each class and this will be output as TYPE in the FRAGSTATS output files. *Enabled* is a

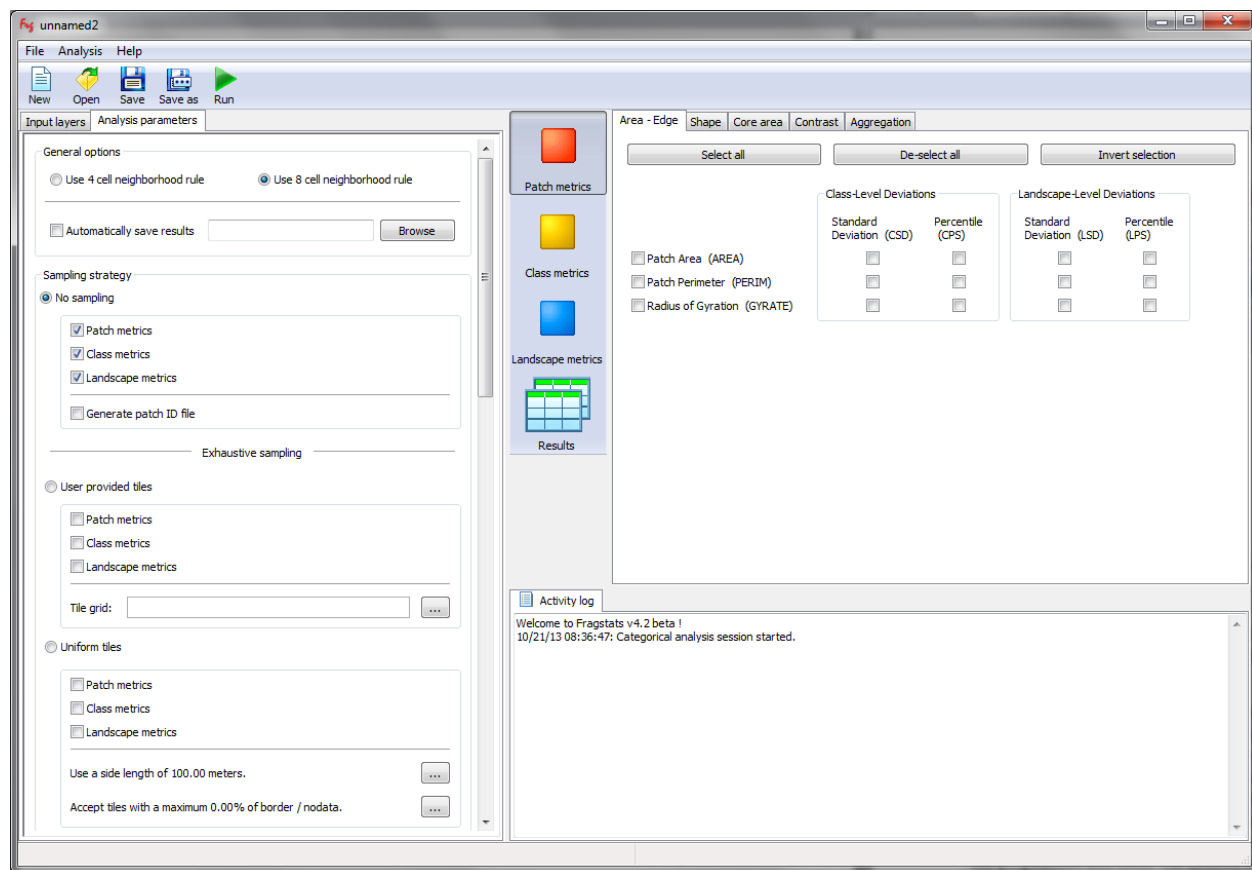
```
ID, Name, Enabled, IsBackground
100,open,true,false
300,resident,true,false
400,water,true,false
500,forest,true,false
600,wetland,true,false
700,urban,true,false
```

logical (true or false) and indicates whether to compute and output statistics for the corresponding class. Lastly, *IsBackground* is logical (true or false) and indicates whether to treat each class as background or not. Note, a "true" involves reclassifying the corresponding class to the background value specified earlier (999 in this case). If we had specified say, 400, as the background value earlier, then *Water* would be treated as background regardless of what is designated in this table.

To use the provided class descriptors file, click on the **Class descriptors Browse** button in the Common tables section of the user interface on the Input layers tab and navigate to the tutorial directory and select the **descriptors.fcd** file.

## 5. Specify additional parameters for the analysis

Next, you need to specify some additional parameters for the analysis. Click on the **Analysis parameters** tab on the left pane of the user interface. Here, is where you chose the neighbor rule for delineating patches (4 cell rule or 8 cell rule) and specify whether you want to sample the landscape to analyze sub-landscapes using one of the alternative sampling schemes (see tutorial #6).



For this tutorial, keep the default 8 cell neighbor rule and the **No sampling** strategy; the exhaustive moving window sampling strategy is covered in tutorial #4 and the rest

of the sampling strategies in tutorial #6. In addition, check the boxes for **Patch**, **Class** and **Landscape** metrics under the No sampling option. Note, you must have at least one of these boxes checked or you will get an error message later when trying to run the model. However, only check the level corresponding to the metrics you want to compute. Some applications will involve patch level metrics, for example when evaluating the spatial character and context of each habitat patch in a metapopulation context. Other applications will involve only the class level metrics, for example when evaluating the fragmentation of a focal class. And still other applications will involve only the landscape level metrics, for example when evaluating overall landscape heterogeneity. Of course, some applications will involve more than one level of metric.

There is an optional check box for generating a patch ID file. If checked, FRAGSTATS will generate a patch ID grid in the same format as the input layer, and each cell will be assigned a unique patch ID value. Thus, all the cells belonging to patch #1 will be assigned the value 1, all cells in patch #2 will be assigned the value 2, and so on. The unique patch ID values will correspond to the unique patch ID values in the PID field of the `basename.patch` output file. In this manner, the patch ID file can be used to connect the patch metric results to the corresponding patch in the landscape. In fact, the `basename.patch` outfile can be joined to the patch ID grid in the GIS if so desired, but we will not illustrate this here.

## 6. Select metrics

Next, you need to select some metrics to compute. Give that you selected patch, class and landscape metrics in step 5, you need to select individual metrics at each of these levels.

To begin, select Patch metrics in the top right pane of the user interface. Click on the **Patch metrics** button and then on each tabbed set of metrics. You can choose a subset of metrics or simply "Select all" -- your choice. Note, on the **Aggregation** tab, if you select either the *Proximity index* or the *Similarity index*, then you also need to specify a Search radius. These metrics are "functional" metrics and thus require additional parameterization. Both of these metrics require a search radius; the *Similarity index* also requires a similarity weights table (see below). To specify a search radius, click on the [...] button and enter the desired search radius in meters; e.g., 500.

Next, click on the **Class metrics** button and then on each tabbed set of metrics. Again, you can choose a subset of metrics or simply "Select all" -- your choice. Note, on the **Area-Edge** tab, if you select *Total Edge* or *Edge Density*, then you need to consider how you want to treat any background or boundary edge in the edge calculations. The default is to not consider any of it as true edge. However, you can choose to treat all of it as edge or any specified proportion as edge. To change the default, click on the [...] button and enter your choice. Note, since the input landscape contains a border and



does not contain any designated background, the issue is mute since we know the true status of every edge segment along the landscape boundary and there are no background edges to worry about. Similarly, on the **Aggregation** tab, if you select the *Connectance index*, then you also need to specify a threshold distance within which patches are deemed "connected". Simply click on the [...] button and enter the desired threshold distance in meters; e.g., 500.

Lastly, click on the **Landscape metrics** button and then on each tabbed set of metrics. Again, you can chose a subset of metrics or simply "Select all" -- your choice. Note, on the **Diversity** tab, if you select *Relative Patch Richness*, then you also need to specify the maximum number of classes (or patch types). Simply click on the [...] button and enter the value; 6 in this case.

## 7. Conditionally, input additional common tables

Next, if you selected any of the Core area metrics, Contrast metrics or Similarity index (on the Aggregation tab) at any level (patch, class or landscape), then you also need to create and input additional ancillary tables in order to parameterize these metrics. If you fail to input these tables or try to input improperly formatted tables, you will get an error message and the analysis will fail. Importantly, it is up to you to create these ancillary files to ensure that they are meaningful to your application. There are no meaningful default values for these tables; creating these tables is how you functionalize the corresponding metrics for your particular application.

Open up the provided **edgedepth.fsq** file in a text editor. Note, FRAGSTATS recognizes .fsq as the extension for these common ancillary tables, but this is not a required extension. This comma-delimited ascii file contains the *depth-of-edge effect* distances (in meters) for each pairwise combination of classes (or patch types). The file must begin with the line: FSQ\_TABLE. It can contain any number of comment lines beginning with the character symbol #. It must contain a class list of literal names (i.e., class descriptors) or numeric class values corresponding exactly to those in the class descriptors file. Note, only one of these lists is required and if both are provided, as in the example below, only the first one encountered is used. Note, the list should include a item for every class in the input grid. If the list contains additional classes not found in the input grid, they are simply ignored. Similarly, if the list omits a class found in the input grid, the edge depths are assumed to be zero by default.

The class list is followed by the edge depths for each pairwise combination of classes, given in the order they are provided in the list, and is read as follows. The row indicates the focal class and the column indicates the adjacent class. Thus, the fourth row is for *Forest* (fourth in the list) as the focal class, and each of the entries represents the depth-of-edge effect distance penetrating into *Forest* from an adjacent class. For example, in the table below, *Open* has an edge effect that penetrates 100 m into *Forest*, *Resident*

penetrates 50 m into *Forest*, and so on. Note, in this example, edge effects are specified only for the *Forest* class.

```
FSQ_TABLE
# comment lines start with # and are allowed anywhere in the table
# literal class names cannot contain spaces
# the order of rows and column is the one specified in the CLASS_LIST_???????
# two types of class lists are allowed CLASS_LIST_LITERAL() and CLASS_LIST_NUMERIC(),
# but only the first one encountered is considered
# class names or ids will be compared with the class descriptors in the model
# and only the matches will be imported
CLASS_LIST_LITERAL(open, resident, water, forest, wetland, urban)
CLASS_LIST_NUMERIC(100, 300, 400, 500, 600, 700)
0,0,0,0,0,0
0,0,0,0,0,0
0,0,0,0,0,0
100,50,50,0,50,200
0,0,0,0,0,0
0,0,0,0,0,0
```

To use the provided edgedepth file, click on the **Edge depth Browse** button in the Common tables section of the user interface on the Input layers tab and navigate to the tutorial directory and select the **edgedepth.fsq** file.

Repeat the process above for the provided **contrast.fsq** and **similarity.fsq** tables; these tables provide the edge contrast weights and similarity coefficients for each pairwise combination of classes (patch types), respectively.

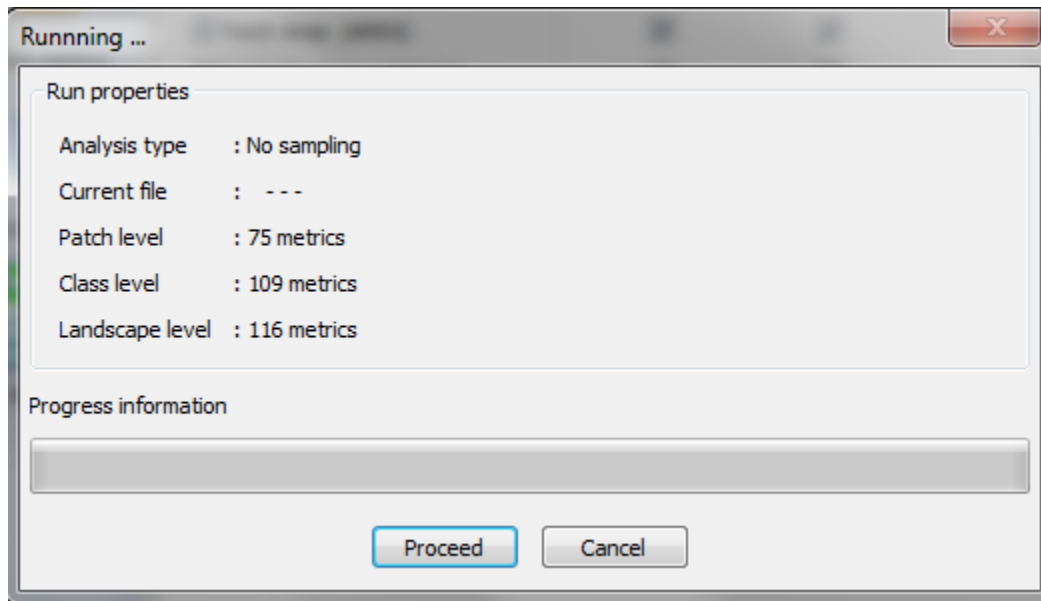
## 8. Optionally, save the model

Next, you might want to save the model for future use. Often times it is easier to open a saved model and modify it than to create a new model from scratch. At any point in the process of parameterizing the model, simply click on the **Save** or **Save as** buttons on the toolbar or select these options from the File drop-down menu to save the model. Simply navigate to the desired directory and enter a file name for the model. Note, FRAGSTATS will automatically assign the extension .fca to the file, which identifies the file as a model for a categorical landscape.

## 9. Run the model

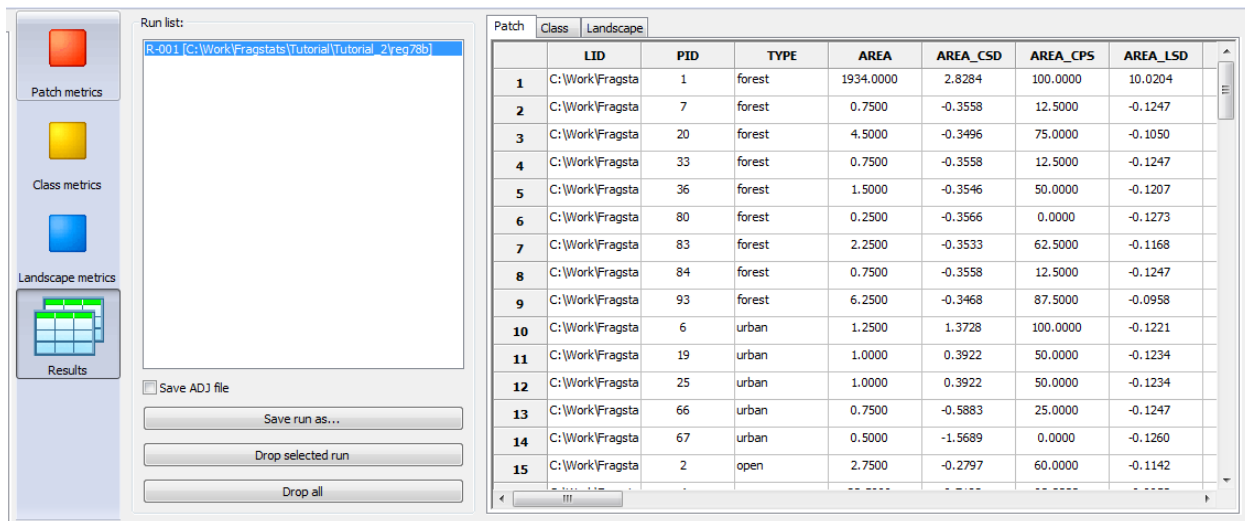
Next, you are ready to run the model. Simply click on the **Run** button on the toolbar or select this option from the Analysis drop-down menu. This will open the Run dialog that lists the analysis type (no sampling in this case), the current file (which gets listed after clicking on the Proceed button, below), and the number of metrics selected at each level (patch, class and landscape). If you like what you see, click on the **Proceed** button, otherwise click on **Cancel** and make the needed modifications to the model. In the

example shown here, the run includes 75 patch metrics, 109 class metrics and 116 landscape metrics because I selected All the available metrics at each level.



## 10. Browse the results

Pay attention to the Activity log in the bottom right pane of the user interface. If all went well, you will learn that the run ended and how long it took. Assuming that the run was successful, you are now ready to browse the results. Click on the **Results** button in the top-right pane of the user interface and then on each tabbed set of results. The results are displayed in a table for each level of metrics selected. So, if you computed patch, class and landscape metrics, as was done here, there will be results in each of the corresponding tabs. Otherwise, only the tab corresponding to the level of metrics computed will contain data; the others will be empty. The **Patch** tab will contain a row for each patch in the input landscape and a column for each patch metric selected. The **Class** tab will contain a row for each (non-background) class in the input landscape and a column for each class metric selected. The **Landscape** tab will contain a single row for the entire input landscape mosaic and a column for each landscape metric selected.



The screenshot shows the Fragstats software interface. On the left is a vertical toolbar with icons for Patch metrics (red square), Class metrics (yellow square), Landscape metrics (blue square), and Results (green grid). The 'Run list' pane shows a single run: 'R-001 [C:\Work\Fragstats\Tutorial\Tutorial\_2\eq78b]'. Below this are buttons for 'Save ADJ file' (checked), 'Save run as...', 'Drop selected run', and 'Drop all'. The main 'Results' table has tabs for 'Patch', 'Class', and 'Landscape'. The 'Landscape' tab is active, displaying a table with 8 columns: LID, PID, TYPE, AREA, AREA\_CSD, AREA\_CPS, and AREA\_LSD. The table contains 15 rows of data for various land use patches.

	LID	PID	TYPE	AREA	AREA_CSD	AREA_CPS	AREA_LSD
1	C:\Work\Fragsta	1	forest	1934.0000	2.8284	100.0000	10.0204
2	C:\Work\Fragsta	7	forest	0.7500	-0.3558	12.5000	-0.1247
3	C:\Work\Fragsta	20	forest	4.5000	-0.3496	75.0000	-0.1050
4	C:\Work\Fragsta	33	forest	0.7500	-0.3558	12.5000	-0.1247
5	C:\Work\Fragsta	36	forest	1.5000	-0.3546	50.0000	-0.1207
6	C:\Work\Fragsta	80	forest	0.2500	-0.3566	0.0000	-0.1273
7	C:\Work\Fragsta	83	forest	2.2500	-0.3533	62.5000	-0.1168
8	C:\Work\Fragsta	84	forest	0.7500	-0.3558	12.5000	-0.1247
9	C:\Work\Fragsta	93	forest	6.2500	-0.3468	87.5000	-0.0958
10	C:\Work\Fragsta	6	urban	1.2500	1.3728	100.0000	-0.1221
11	C:\Work\Fragsta	19	urban	1.0000	0.3922	50.0000	-0.1234
12	C:\Work\Fragsta	25	urban	1.0000	0.3922	50.0000	-0.1234
13	C:\Work\Fragsta	66	urban	0.7500	-0.5883	25.0000	-0.1247
14	C:\Work\Fragsta	67	urban	0.5000	-1.5689	0.0000	-0.1260
15	C:\Work\Fragsta	2	open	2.7500	-0.2797	60.0000	-0.1142

Once you have verified that the tables contain results, the next step is probably to export the tables so that you can use them in subsequent analyses. To export the results, simply click on the desired run in the Run list (note, at this point you should have only one run listed), and then click on **Save run as...** and navigate to the desired folder and enter a **basename** for the output files. The basename will be given an extension corresponding to each level of metrics computed (basename.patch, basename.class, and basename.land). Thus, in this example, three files will be created corresponding to the three levels of metrics computed. Note, a fourth output file containing the cell adjacency information can be created by checking the **Save ADJ file** check box (basename.adj). Each of these output files is a comma-delimited ascii file and can easily imported into other software such as R and Excel for further analysis and summary.

Note that is also possible to automatically save the results with the execution by checking the **Automatically save results** option on the Analysis parameters tab in the left pane of the user interface. Simply check this box and then click on the **Browse** button to navigate to a desired output directory and enter a basename for the output file(s).

## Tutorial 3: Batch processing multiple grids

In this tutorial, you will batch process multiple input grids. Note, this tutorial assumes that you now have a basic working understanding of FRAGSTATS from completing the first two tutorials and/or reading the detailed user guidelines that come with the FRAGSTATS software.

### 1. Open FRAGSTATS

First, open FRAGSTATS from the start menu or by double clicking on the FRAGSTATS icon on the desktop. Remember, if you have an ESRI ArcGIS (**v10.0 or earlier**) installation, there may be a significant delay (up to 30 seconds in some cases) in opening the user interface as FRAGSTATS tries to secure a valid ArcGIS license, so be patient and don't forget to thank ESRI for their license management while you're waiting. Note, if you have a later version of ArcGIS installed on your machine or you don't have a valid ESRI Spatial Analyst license or you simply don't want to run FRAGSTATS with ArcGrids, then be sure that your system's environmental "path" variable does **NOT** contain a path to ArcGIS. See tutorial #1 for details.

### 2. Open a FRAGSTATS model

Next, open a saved FRAGSTATS model. If you saved the model from tutorial #2, you can open that model (if is not already open). If you didn't save the model, open the provided model by clicking on the **Open** button on the toolbar or select **Open** from the File drop-down menu and navigate to the tutorial #3 directory and select the file named **fragmodel.fca**.

### 3. Create and input a batch file

Next, so that we can start fresh, remove any layers that are already loaded into the model. Specifically, click on **Remove all layers** in the Batch management section of the user interface on the Input layers tab. This will delete all grids previously loaded into the model.

Before you can import a batch file, you need to first create one. A batch file is a comma-delimited ascii file that lists the input grids to be analyzed according to the same model parameterization. Specifically, the batch file contains the full path and name of each input file and a description of the grid attributes associated with each input grid, including the cell size, background value, number of rows and columns, band number, and data type.

**GeoTIFFs**--Open the provided **geotiffbatch.fbt** file in a text editor. Note, FRAGSTATS recognizes .fbt as the extension for batch files, but this is not a required extension.

```
C:\Work\Fragstats\Tutorial\Tutorial_3\reg21b.tif, x, 999, x, x, 1, x, IDF_GeoTIFF
C:\Work\Fragstats\Tutorial\Tutorial_3\reg66b.tif, x, 999, x, x, 1, x, IDF_GeoTIFF
C:\Work\Fragstats\Tutorial\Tutorial_3\reg78b.tif, x, 999, x, x, 1, x, IDF_GeoTIFF
```

This file contains three rows, one for each input grid to analyze, and the following eight fields:

1. *Input grid*: The first field contains the full path and name of the input grid. Note, if we were using ArcGrids we would put the name of the grid folder in place of the grid name since the grid is actually a folder containing several files.
2. *Cell size*: The second field contains an integer value corresponding to the cell size (in meters).
3. *Background value*: The third field contains an integer value corresponding to the designated background value. Note, any class designated as background in the Class descriptors file (see tutorial #2) will be reclassified to this class value and treated as background.
4. *Number of rows*: The fourth field contains an integer value corresponding to the number of rows in the input image.
5. *Number of columns*: The fifth field contains an integer value corresponding to the number of columns in the input image.
6. *Band number*: The sixth field contains an integer value corresponding to the band number to interpret in the input image, which by default is #1 but can vary for some of the input data formats.
7. *Nodata value*: The seventh field contains an integer value corresponding to the nodata value.
8. *Input data format*: The last field contains a character string identifying the input data format, with the following options corresponding to the various input data format types: (e.g., IDF\_GeoTIFF, IDF\_ASCII, IDF\_8BIT, IDF\_ARCGRID, etc.). Note, in this example, since the input grids are all GeoTIFFs, the grid attributes associated with cell size, number of rows and columns, and nodata value are not needed; an "x" is used in place of the argument.

To use the provided batch file, click on the **Import batch** button in the Batch management section of the user interface on the Input layers tab and navigate to the tutorial directory and select the **geotiffbatch.fbt** file.

**Ascii grids (and other formats)**--If you are working with ascii grids, open the provided **asciibatch.fbt** file in a text editor. Note, a batch file and the corresponding input grids are not provided for other input data formats, but it would be similar to the

one shown here except the last argument that identifies the input data format would reflect the input data format (e.g., IDF\_8BIT, IDF\_GeoTIFF, etc.).

```
C:\Work\Fragstats\Tutorial\Tutorial_3\reg21b.asc, 50, 999, 102, 102, 1, 9999, IDF_ASCII  
C:\Work\Fragstats\Tutorial\Tutorial_3\reg66b.asc, 50, 999, 102, 102, 1, 9999, IDF_ASCII  
C:\Work\Fragstats\Tutorial\Tutorial_3\reg78b.asc, 50, 999, 102, 102, 1, 9999, IDF_ASCII
```

Note that with ascii files, you must specify all of the grid attributes associated with cell size, number of rows and columns, and nodata value.

To use the provided ascii batch file, click on the **Import batch** button in the Batch management section of the user interface on the Input layers tab and navigate to the tutorial directory and select the **asciibatch.fbt** file.

#### 4. Complete the model parameterization

Next, complete the model parameterization. Running 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 analysis parameters and select and parameterize the individual metrics, as described in tutorial #2. If you haven't already completed the model parameterization to your satisfaction, do so now (see tutorial #2 for help if you need it) or simply go with the model as parameterized in the provided model.

#### 5. Run the model and browse the results

Lastly, you are ready to **Run** the model and **Browse** the results, as before (see tutorial #2). The only notable difference between this run and the previous run on a single landscape (tutorial #2) is that the **Run list** in the top-right pane of the user interface now contains a list of outputs associated with this run. Specifically, the Run list contains a separate set of results for each of the input grids in the batch file. Click on each of the list items to view the corresponding patch, class and landscape results. To save the results, click on Save run as... and navigate to the desired directory and enter a basename for the output files. Note, the basename.patch file will contain all of the results for all of the input grids associated with this run. In this case, the file will contain the patch metrics for all of the patches in each of the three input grids. Similarly, the class and landscape files each will contain all of the results appended into a single file.



## Tutorial 4: Moving window analysis

In this tutorial, you will quantify local landscape structure gradients via a moving window analysis on a single input grid. Briefly, a moving window analysis places a window (a local kernel of user-specified shape and size) over each focal cell and computes the selected metric and returns the metric value back to the focal cell. Thus, each window around a focal cell is treated like a sub-landscape. By repeating this process for every cell, the end result is a surface in which the height of the surface at each cell is equal to the value of the metric. The surface represents a gradient in local landscape structure, and there is a separate surface for each metric selected.

Note, this tutorial assumes that you now have a basic working understanding of FRAGSTATS from completing tutorials #1 and #2 and/or reading the detailed user guidelines that comes with the FRAGSTATS software.

### 1. Open FRAGSTATS

First, open FRAGSTATS from the start menu or by double clicking on the FRAGSTATS icon on the desktop. Remember, if you have an ESRI ArcGIS (**v10.0 or earlier**) installation, there may be a significant delay (up to 30 seconds in some cases) in opening the user interface as FRAGSTATS tries to secure a valid ArcGIS license, so be patient and don't forget to thank ESRI for their license management while you're waiting. Note, if you have a later version of ArcGIS installed on your machine or you don't have a valid ESRI Spatial Analyst license or you simply don't want to run FRAGSTATS with ArcGrids, then be sure that your system's environmental "path" variable does **NOT** contain a path to ArcGIS. See tutorial #1 for details.

### 2. Create a FRAGSTATS model

Next, create a New FRAGSTATS model, as before (see tutorial #2). Simply click on the **New** button on the tool bar or select **New** from the File drop-down menu. This creates a blank model for you to parameterize. Or, if you prefer, you can open the provided model (**fragmodelMw.fca**) -- but really you shouldn't, since this is a tutorial after all, and the best way of learning is by doing. Note, if you do open the provided model, you will likely need to modify the path to the tutorial grid and the corresponding class descriptors file (see tutorial #2 for instructions).

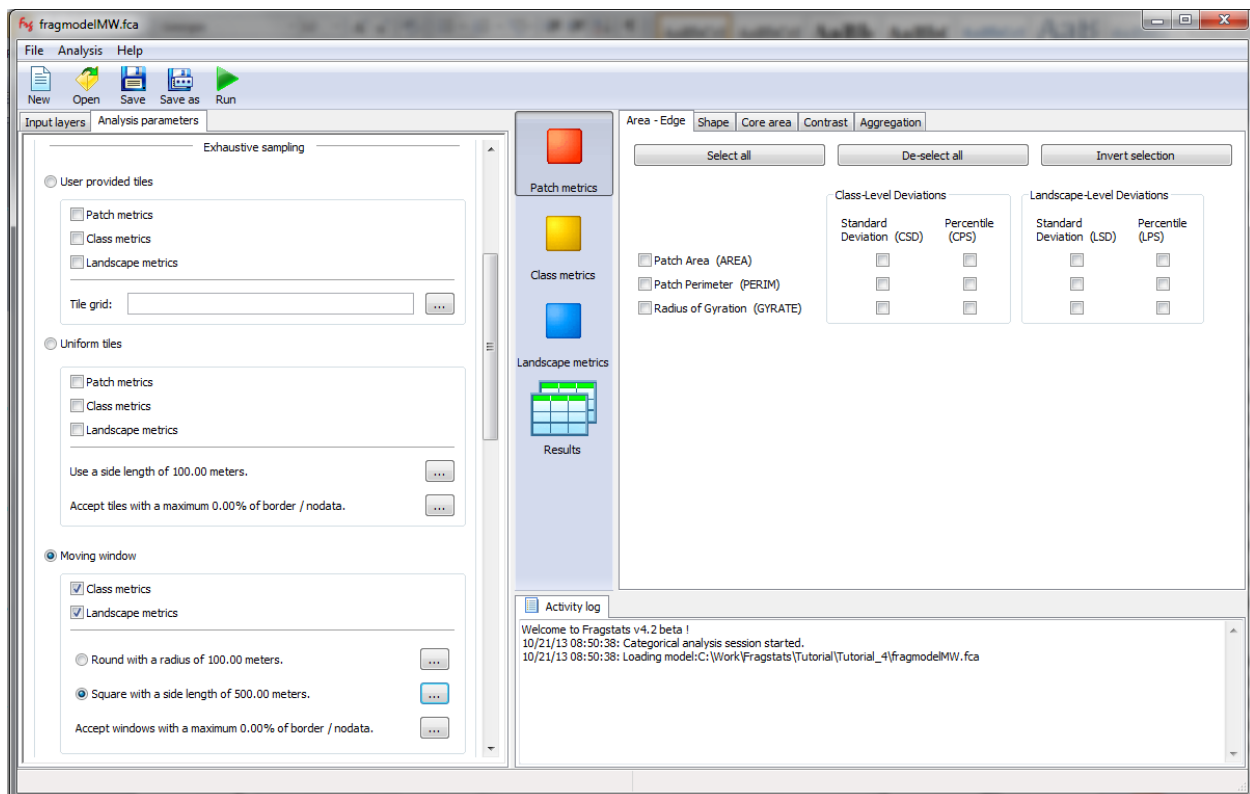
### 3. Import a grid

Next, import a grid to analyze, as before (see tutorial #2). Note, if you are using the provided model (**fragmodelMw.fca**), you will probably have to change the path to the layer already loaded, since you are likely to have a different path to the tutorial on your machine. In this case, simply click on the **Edit layer** button in the Batch management

section of the user interface on the Input layers tab and navigate to the **reg78b.tif** dataset on your machine. If you are starting from scratch, click on the **Add layer** button in the Batch management section of the user interface on the Input layers tab to open the import data dialog and add the provided **reg78b.tif** grid. Note, if you are working with ESRI ArcGIS, import the ArcGrid; otherwise, import either the corresponding ascii or binary grid(16 or 32 bit).

## 4. Specify additional parameters for the analysis

Next, you need to specify some additional parameters for the analysis. Click on the **Analysis parameters** tab on the left pane of the user interface. Here, is where you chose the neighbor rule for delineating patches (4 cell rule or 8 cell rule) and specify whether you want to sample the landscape to analyze sub-landscapes and, if so, by which method.



For this tutorial, keep the default 8 cell neighbor rule and select the Exhaustive sampling **Moving window** option. In addition, check the boxes for **class** and **landscape metrics**, and choose between the **Round** or **Square** local kernel. Next, click on the [...] button associated with the chosen kernel and enter **500** (in meters) as either the radius of a circular kernel or the side of a square kernel. Lastly, leave the default maximum 0% of border/nodata to accept in the window. With this option set to 0%, any window containing any border (negative cell values) or nodata will be disregarded and the focal cell value set to nodata in the output grid. Note, this prevents

partial windows from being analyzed. If you want to analyze every window, regardless of the percentage comprised of border/nodata, then click on the [...] button and change this threshold to 100%; but be aware of the implications for the computed metrics since the total landscape area (i.e., window area) will vary among windows.

## 5. Modify the class descriptors table and import

Because the moving window analysis is quite compute intensive, it can take a very long time to complete on a large landscape. In addition, because each metric selected will produce a separate grid, it is prudent to be extremely selective in the choice of metrics (see below) and carefully consider which landcover class or classes to focus on (for class level metrics). For the purpose of this tutorial, we will focus solely on the *Forest* landcover class.

To restrict the moving window analysis to the *Forest* landcover class for the class level metrics you need to modify the class descriptors table. Open up the provided **descriptors.fcd** file in a text editor and change the Enabled argument to "false" for all the classes except *Forest*, as shown here.

```
ID, Name, Enabled, IsBackground
100,open,false,false
300,resident,false,false
400,water,false,false
500,forest,true,false
600,wetland,false,false
700,urban,false,false
```

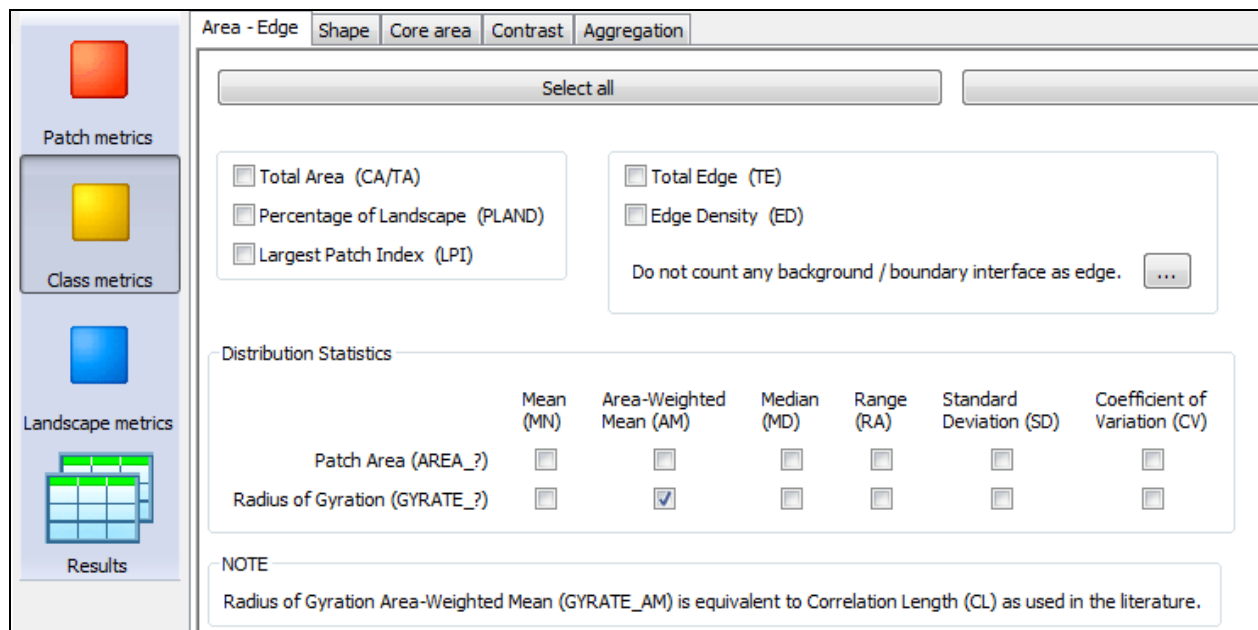
You can save this modified file to the same file or choose a different file name. If you choose a different file name be sure to import the correct file in the next step.

Next, click on the **Class descriptors Browse** button in the Common tables section of the user interface on the Input layers tab and navigate to the tutorial directory and select the modified **descriptors.fcd** file (be sure to select the modified file you saved, or use the one provided, **descriptors.modified.fcd**).

## 6. Select metrics

Next, you need to select some metrics to compute. Give that you selected Class and Landscape metrics in step 5, you need to select one or more metrics at each of these levels.

Click on the **Class metrics** button in the top right pane of the user interface and then on the **Area-Edge** tab (if it is not already the active tab). Check the box for the *Area-Weighted Mean (AM) Radius of Gyration* (also known as Correlation length). This is a measure of the physical continuity of the landscape and is often used in studies on habitat fragmentation.



Click on the **Landscape metrics** button in the top right pane of the user interface and then on the **Area-Edge** tab (if it is not already the active tab). Check the box for the *Area-Weighted Mean (AM) Radius of Gyration*.

Note, you have selected a single metric (Correlation length), but computed both at the class level for the *Forest* class only and at the landscape level for the entire patch mosaic. Select other metrics if you wish; each metric will produce an output grid.

## 7. Run the model

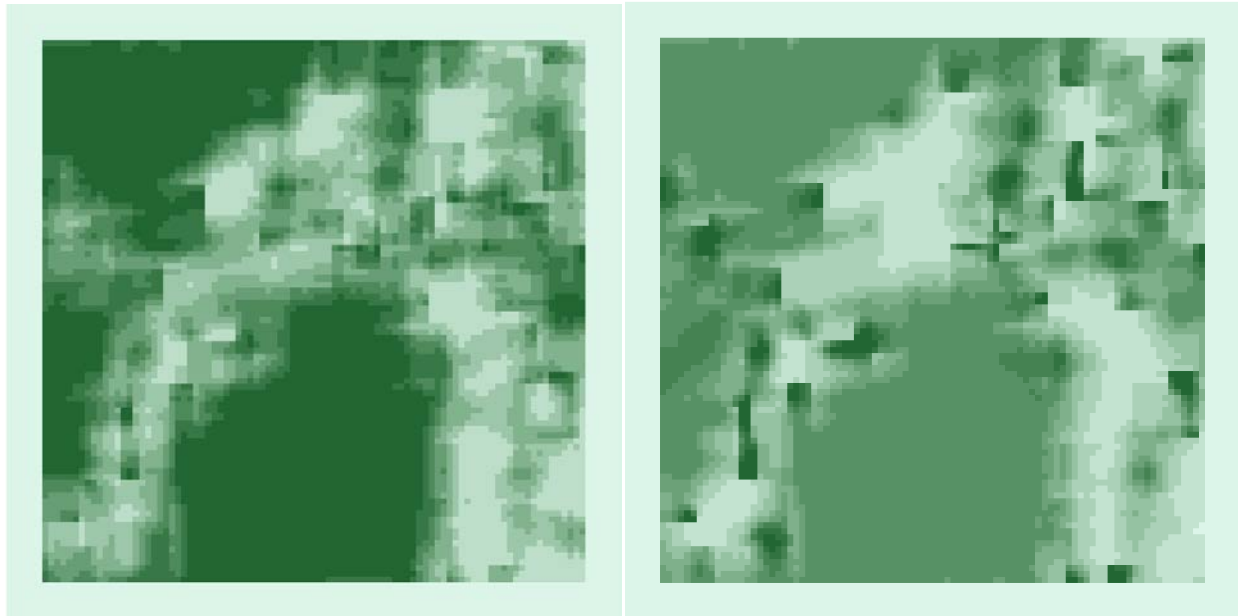
Lastly, you are ready to **Run** the mode, as before (see tutorial #2). The major difference between this run and the runs from the previous tutorials is that the **Run list** in the top-right pane of the user interface is going to be empty after the run is complete because the moving window analysis produces grids as output instead of tables. The other notable difference is that the run takes much longer. Instead of computing the metrics for a single input grid, in this example, the metrics are computed for 6,561 separate windows (sub-landscapes), one for each cell in the landscape in which the specified window (500-m radius circle in this case) does not extend beyond the landscape boundary; i.e., the entire window contains positively-valued cells.

## 8. View the results

Lastly, once the run is complete, you can view the results. In this case, the results are grids in the same data format as the input grid.

**GeoTIFF/ArcGIS grids.**--If you analyzed a GeoTIFF or ESRI ArcGrid, open up the provided **fragtutorial\_4.mxd** project in ArcMap. The project contains the reg78b grid

preloaded, as described in tutorial #1, along with the *Area-weighted mean radius of gyration* at the landscape level (**gyrate\_am**) and at the class level for the forest class (**gyrate\_am\_500**). If you computed any other metrics, you can add them to the project now. Specifically, you should now have a folder named **reg78b\_MW1** (first moving window analysis on reg78b) in the Tutorial\_4 directory and inside this folder there should be grids for each of the metrics selected. Toggle the gyrate\_am.tif and gyrate\_am\_500.tif layers on an off with the landcover grid in the background.



The first thing you should notice is that both grids contain a nodata border that is equal to the radius of the window used, which was 500 m in this case. FRAGSTATS does not compute focal cell values for cells within the specified window radius of the edge of the rectangular input grid, or within the specified window radius of the landscape boundary (i.e., border/nodata), because too many of the metrics are sensitive to landscape extent. If you want moving window results for the entire landscape, the only unbiased solution is to expand the landscape to include an appropriately sized buffer so that the end result is real values for all cells within the landscape of interest.

The second thing you should notice is that the window shape and size is evident in the results. This is not always so obvious and depends largely on the metric. In this case, the area-weighted mean radius of gyration is particularly sensitive to the exact pattern of what gets included in the window. Try running the analysis again but with a variety of other metrics to see how they vary.

The last thing you should notice is that the result differs, but only subtly, between the landscape (left figure above) and class (right figure above) grids. This is because the Forest class is dominant in this particular landscape, so the extensiveness of Forest (as measured by this metric) dominates the pattern at both the class and landscape level.

**Ascii/binary grids.**--If you analyzed ascii or binary grids, it is a bit more difficult to view the moving window results without importing them into your favorite GIS. However, if you are an R user, you can use the following script (or open the provided script, **tutorial\_4.R**) to plot the grid in R. Note, there are several ways to plot the grids in R. Here, we will use the Raster package, which you will need to download and install if you don't already have it. Try the following script:

First, load the Raster library:

```
library(raster)
```

Next, set the working directory to wherever you have installed the tutorial; e.g.:

```
setwd('c:/work/fragstats/tutorial/tutorial_4/ reg78b.asc_mw1')
```

Next, read in the ascii grid, as a matrix, into an object (m), but also specify the negative background value (-999 by default) to treat as nodata so that the border that was created from the moving window analysis is not read in as real data:

```
m<-as.matrix(read.table('gyrate_am.asc',na.strings='-999'))
```

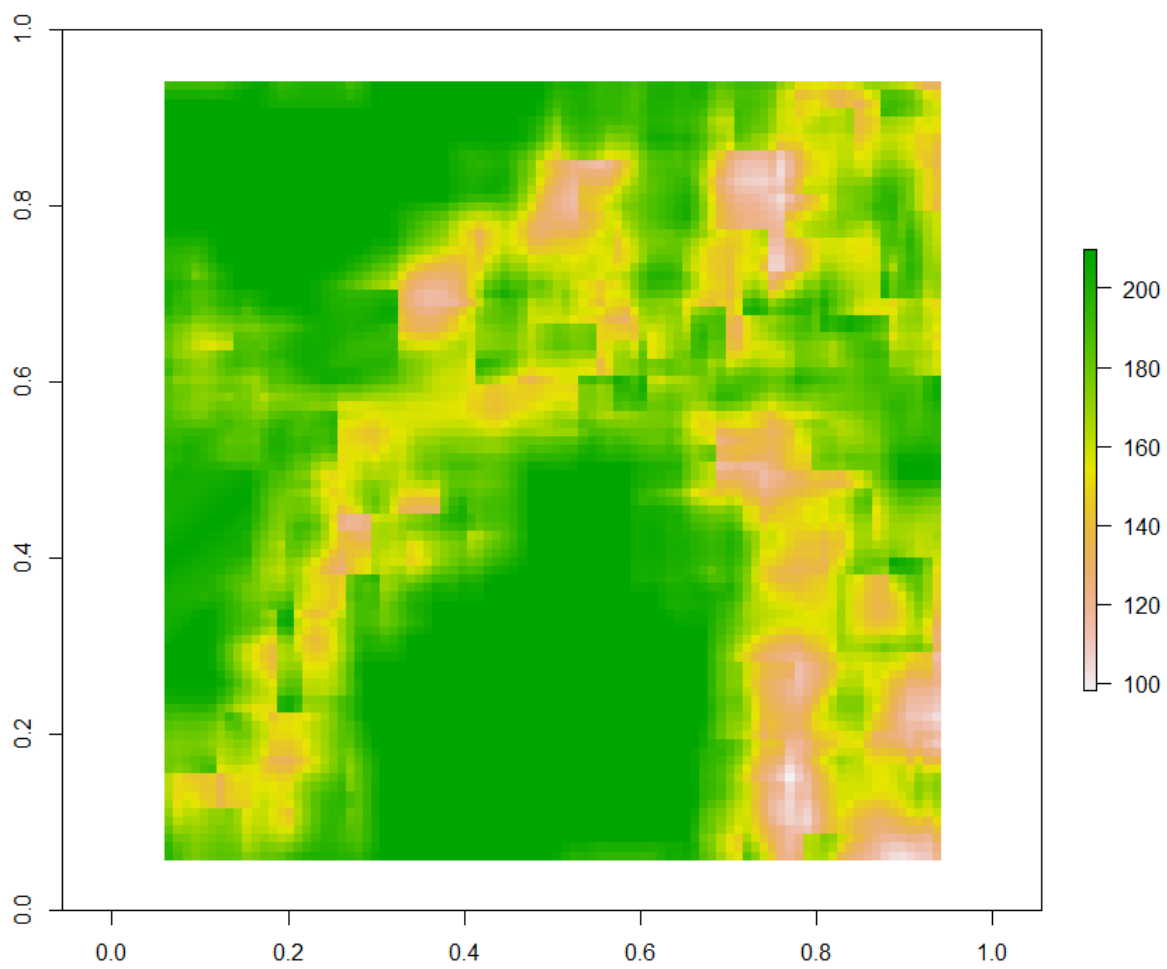
Next, convert the matrix object (m) into a Raster object (also assigned to m):

```
m<-raster(m)
```

Finally, plot the image:

```
plot(m)
```

If you want, plot the class level metric using the previous script, but substituting the grid name: `gyrate_am_500.asc`.





## Tutorial 5: Running the command line version from R

In this tutorial, you will run the command line version of FRAGSTATS from R:

*R Development Core Team (2008). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL <http://www.R-project.org>.*

The command line version can be run directly from the system command line as per the instructions in the complete user guidelines. However, it can also be called from within other programs such as R, as illustrated in this tutorial.

This tutorial assumes that you now have a basic working understanding of FRAGSTATS from completing tutorials #1 and #2 and/or reading the detailed user guidelines that comes with the FRAGSTATS software. In particular, this tutorial assumes that you know how to build a FRAGSTATS model via the graphical user interface (see tutorial #2). In addition, this tutorial assumes that you already are familiar with R and have it installed on your machine and know how to work in R.

### 1. Create a FRAGSTATS model

Currently, the command line version of FRAGSTATS requires that a model first be created via the graphical user interface version, as illustrated in tutorial #2. Eventually, we would like to build a mechanism for parameterizing the model without having to open the user interface, but for the time being you will have to be satisfied with using the user interface to parameterize a model.

So, the first step is to open the user interface and parameterize the model. See tutorial #2 for a detailed example. For this tutorial, we have provided a ready-to-use model (**fragmodelR.fca**). The most important difference between this model and the one created and saved in tutorial #2 is that this model has no grid layers added to the batch manager and it does not have a basename for the output files specified and the "automatically save results" box checked. Note, even if you have grid layers loaded in the batch manager and/or have a basename for the output files specified and the "automatically save results" box checked, you can override these model specifications with the use of the command line switches, as illustrated below.

### 2. Open R and run the script provided

Next, open R and work through the script below (or open the provided script, **tutorial-5.R**).

First, set the working directory to wherever you have installed the tutorial; e.g.:

```
setwd('c:/work/fragstats/tutorial/tutorial_5')
```

Next, create a FRAGSTATS batch file to list the input grids (or single grid in this case) to be analyzed along with their grid attributes. First, create a temporary object containing the contents of the batch file, and then write it to disk:

```
temp<-paste('c:\\work\\fragstats\\tutorial\\tutorial_5\\  
reg78b.tif','x','999','x','x',1,'x','IDF_GeoTIFF',sep=',')  
  
write.table(temp,file='geotiffbatch.fbt',quote=FALSE,row.names=FALSE,  
col.names=FALSE)
```

There are a few details about the above script worth noting:

- First, the first argument of the FRAGSTATS batch file is the full path and name of the input grid. To get R to output the path with the conventional backslashes, you need to put double backslashes in the script.
- Second, the second, fourth, fifth and seventh arguments of the batch file pertain to the cell size, number of rows, number of columns and nodata value which do not need to be specified for a GeoTIFF and thus the arguments are set to 'x'. If you working with ascii or binary grids you will need to include values for all of the arguments (see below).
- Third, note the use of the **sep=','** argument, which designates the delimiter to be used between items, which needs to be a comma in a FRAGSTATS batch file.
- Lastly, note the use of the **quote=FALSE**, **row.names=FALSE** and **col.names=FALSE** arguments in the write.table () function, which are necessary to ensure the omission of quotes and row and column headers in the output file.

Next, use the system() function to execute the FRAGSTATS command line executable (frg.exe) from the system command line:

```
system('frg -m fragmodelR.fca -b geotiffbatch.fbt -o  
c:\\work\\fragstats\\tutorial\\tutorial_5\\fragout')
```

There are a few details about the script above worth noting:

- First, the **system ()** function is simply a generic function for accessing the operating system command line.
- Second, the **-m** switch is *required* and must be followed by a valid FRAGSTATS model. In this case, we have specified the provided model (fragmodelR.fca) but it can be any valid model that you have created using the user interface.
- Third, the **-b** switch is *optional* and is used to supply a properly formatted FRAGSTATS batch file. In this case, we created the batch file in R. However, we

could have easily created the batch file outside of R using any text editor and simply specified that file here. Note, because we specified a batch file here using the -b switch, it was not necessary to load any grid layers into the batch manager in the model via the user interface. Thus, the fragmodelR.fca provided does not have any grids added to the batch manager. However, if the model did contain layers already added to the batch manager, they would be ignored; the -b switch overrides any layers added to the model specified.

- Fourth, the -o switch is *optional* and is used to supply a basename and path for the output files. Note, the basename must include a full path to an existing folder and the basename to be given to the output files. As described above, we need to use double backslashes to produce a path with the conventional single backslash. Note, if the -o switch is not specified, then the model must have the "automatically save results" box checked and a full path to an existing folder and the basename to be given to the output files specified.

Lastly, read in the FRAGSTATS output generated from the execution above. In this case, the model included patch, class, and landscape metrics and so you have three output files:

```
frag.patch<-read.csv('fragout.patch',header=TRUE,strip.white=TRUE,  
na.strings='N/A')
```

```
frag.class<-read.csv('fragout.class',header=TRUE,strip.white=TRUE,  
na.strings='N/A')
```

```
frag.land<-read.csv('fragout.land',header=TRUE,strip.white=TRUE,  
na.strings='N/A')
```

There are a few details about the above script worth noting:

- First, the filename of the file to read, e.g., 'fragout.patch' must match the basename supplied in either the system call using the -o argument (see above; in this case the basename specified was 'fragout') or in the filename specified in the model (fragmodelR.fca) if the 'automatically save results' box was checked. Note, the extensions '.patch', '.class', and '.land' are added automatically by the software.
- Second, note that we used the **header=TRUE** argument because the FRAGSTATS output files contain a header line.
- Lastly, note that we also used the **strip.white=TRUE** argument to delete any leading and trailing blank spaces in the FRAGSTATS output files (mainly for cosmetic reasons).

Now that you have R objects containing the patch, class and landscape results, you can work with the output in R as you please. Of course the real advantage of R is running FRAGSTATS on multiple landscapes. This can be done as a single batch file or each landscape can be run separately and the outputs appended together. In either case, the resulting tables in R can be sorted, aggregated, plotted, merged with other data and incorporated into statistical models, etc.

As an example, you can combine the landscape and class metrics into a single dataframe in wide format using the `frag.combine()` function in the Rfrag library available from the Fragstats website. First, you will have to install the Rfrag library. In RStudio you can install packages from the Tools drop-down menu. In this case, change the "Install from" to "Package Archive File", navigate to the folder where you stored the Rfrag.zip file, and then click on the "Install" button. Once you have successfully installed the Rfrag package be sure to load it into memory as follows:

```
library(Rfrag)
```

Now you are ready to run the `frag.combine()` function as follows:

```
frag.combine('c:/work/fragstats/tutorial/tutorial_5/',inland='fragout.land',  
            inclass='fragout.class')
```

This function takes corresponding FRAGSTATS landscape (.land) and class (.class) output files and combines them into a single wide format file with a single row for each unique input landscape (LID). The class metrics are assigned names by combining the class name (TYPE) with each class metric and assigned to a column. Thus, a single landscape containing 6 classes and 4 class metrics, as in this example, will produce a data frame containing 24 columns (variables) for the factorial combination of classes and class metrics, and these will be added to the columns containing the landscape metrics, which in this example consists of 6 variables, for a total of 30 columns (variables). Note, if the input files contain multiple landscapes (e.g., resulting from running a batch file), the output file will contain a single row for each input landscape.

**Ascii/binary grids.**--If you are working with ascii or binary grids, you will need to modify the script above slightly. In a typical application you are probably going to be generating your own grids in R, and thus you will not need to do this next step. However, for this tutorial, read in the provided ascii grid (reg78b.asc), as a matrix, into an object (m):

```
m<-as.matrix(read.table('reg78b.asc'))
```

Next, write out the ascii grid to disk. Note, you may not need to do this step in a real application; it all depends on how you are generating the grids to be analyzed in

FRAGSTATS. But for this tutorial, write out the grid you just read in back to disk, noting that you don't want to write out the row and column names of the matrix:

```
write.table(m,file='reg78b.asc',row.names=FALSE,col.names=FALSE)
```

Next, create a FRAGSTATS batch file to list the input grids (or single grid in this case) to be analyzed along with their grid attributes. First, create a temporary object containing the contents of the batch file, and then write it to disk:

```
temp<-paste('c:\\work\\fragstats\\tutorial\\tutorial_5\\ reg78b.asc',  
            '50','999',dim(m)[1],dim(m)[2],'9999','IDF_ASCII',sep=',')
```

```
write.table(temp,file='asciibatch.fbt',quote=FALSE,row.names=FALSE,  
            col.names=FALSE)
```

The fourth and fifth arguments of the batch file pertain to the number of rows and columns, and we extracted these attributes from the matrix object using the `dim()` function, which returns a vector with the matrix dimensions. We used **`dim(m)[1]`** to extract the first element of the vector, which is equal to the number of rows, and **`dim(m)[2]`** to extract the second element of the vector, which is equal to the number of columns.

Next, use the `system()` function to execute the FRAGSTATS command line executable (`frg.exe`) from the system command line:

```
system('frg -m fragmodelR.fca -b asciibatch.fbt  
-o c:\\work\\fragstats\\tutorial\\tutorial_5\\fragout')
```

Lastly, read in the FRAGSTATS output generated from the execution above:

```
frag.patch<-read.csv('fragout.patch',header=TRUE,strip.white=TRUE,  
                    na.strings='N/A')
```

```
frag.class<-read.csv('fragout.class',header=TRUE,strip.white=TRUE,  
                    na.strings='N/A')
```

```
frag.land<-read.csv('fragout.land',header=TRUE,strip.white=TRUE,  
                   na.strings='N/A')
```

## Tutorial 6. Using sampling strategies to analyze sub-landscapes

In this tutorial, you will use various sampling strategies to analyze sub-landscapes, including: 1) exhaustive sampling based on either user-provided tiles, or a systematic tiling scheme, and 2) partial sampling based on user-specified windows around either user-provided or random sample points.

Note, this tutorial assumes that you now have a basic working understanding of FRAGSTATS from completing tutorials #1 and #2 and/or reading the detailed user guidelines that comes with the FRAGSTATS software.

### 1. Open FRAGSTATS

First, open FRAGSTATS from the start menu or by double clicking on the FRAGSTATS icon on the desktop. Remember, if you have an ESRI ArcGIS (**v10.0 or earlier**) installation, there may be a significant delay (up to 30 seconds in some cases) in opening the user interface as FRAGSTATS tries to secure a valid ArcGIS license, so be patient and don't forget to thank ESRI for their license management while you're waiting. Note, if you have a later version of ArcGIS installed on your machine or you don't have a valid ESRI Spatial Analyst license or you simply don't want to run FRAGSTATS with ArcGrids, then be sure that your system's environmental "path" variable does **NOT** contain a path to ArcGIS. See tutorial #1 for details.

### 2. Create a FRAGSTATS model

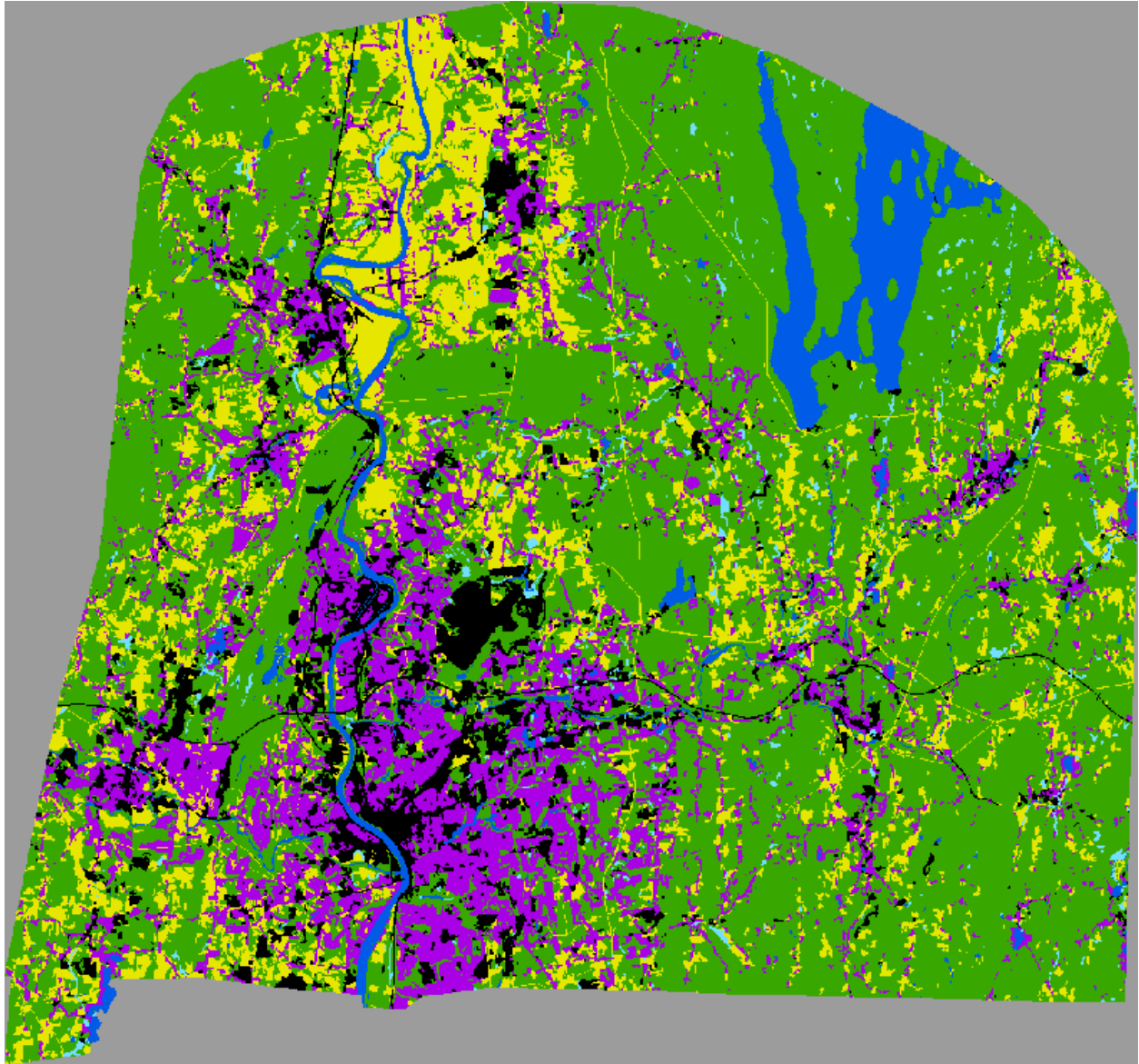
Next, create a New FRAGSTATS model, as before (see tutorial #2). Simply click on the **New** button on the tool bar or select **New** from the File drop-down menu. This creates a blank model for you to parameterize.

### 3. Import a grid

Next, import a grid to analyze, as before (see tutorial #2). Specifically, click on the **Add layer** button in the Batch management section of the user interface on the Input layers tab to open the import data dialog and add the provided **lugrid** grid. Note, if you are working with GeoTIFFs (or ArcGrid), simply import the GeoTIFF (or ArcGrid); otherwise, import the corresponding ascii grid with the following grid attributes:

Row count: 1035  
Column count: 1104  
Background value: 999  
Cell size: 50  
Nodata value: 9999

Let's recall from tutorial #1 what the input grid looks like:



If you are working with ESRI ArcGIS, open up the provided **fragtutorial\_6.mxd** project in ArcMap. The project contains several data layers, as listed in the table of contents, including a landcover grid (**lugrid.tif**) for an arbitrary extent in western Massachusetts, and a tile grid (**tiles.tif**) and points grid (**points.tif**) as described below.

If you are not working with ESRI ArcGIS, you can open up your GIS software and load the GeoTIFFs, or you can use R to view the ascii grids provided using the procedures outlined in tutorial #1, but modifying the script accordingly for the ascii grids provided (**lugrid.asc**, **tiles.asc**, and **points.asc**).



## 4. Optionally, input common tables

Next, you have the option of inputting a class descriptors table and other common tables, depending on the intended choice of metrics, as before (see tutorial #2). Recall, the class descriptors table allows you to specify a character description (i.e., patch type) for each numeric class value, specify whether to compute statistics for each class, and whether to designate each class as background. The class descriptors table is optional. If you do not provide this table, then the numeric class values are used in the output, all classes are enabled and none are treated as background except any class with the assigned background value (999 in this case).

To use the provided class descriptors file, click on the **Class descriptors Browse** button in the Common tables section of the user interface on the Input layers tab and navigate to the tutorial directory and select the **descriptors.fcd** file.

Similarly, if you intend to select any of the Core area metrics, Contrast metrics or Similarity index (on the Aggregation tab) at any level (patch, class or landscape), then you also need to create and input additional ancillary tables in order to parameterize these metrics. Recall, if you fail to input these tables or try to input improperly formatted tables, you will get an error message and the analysis will fail. To use the provided edgedepth file, click on the **Edge depth Browse** button in the Common tables section of the user interface on the Input layers tab and navigate to the tutorial directory and select the **edgedepth.fsq** file. Repeat the process above for the provided **contrast.fsq** and **similarity.fsq** tables, as appropriate; these tables provide the edge contrast weights and similarity coefficients for each pairwise combination of classes (patch types), respectively.

## 5. Select metrics

Next, you need to select some metrics to compute, as before (see tutorial #2). Normally, as in the previous tutorials, we would specify the additional parameters for the analysis prior to selecting metrics, but the order of operations does not matter and for the purpose of this tutorial it is more convenient to select metrics first and then work through the various sampling methods. And for our purposes, let's focus the analysis on class- and Landscape-level heterogeneity.

To begin, click on the **Class metrics** button and then on each tabbed set of metrics. You can choose a subset of metrics or simply "Select all" -- your choice. Note, on the **Area-Edge** tab, if you select *Total Edge* or *Edge Density*, then you need to consider how you want to treat any background or boundary edge in the edge calculations. The default is to not consider any of it as true edge. However, you can choose to treat all of it as edge or any specified proportion as edge. To change the default, click on the [...] button and enter your choice. Note, since the input landscape contains a border and

does not contain any designated background, the issue is mute since we know the true status of every edge segment along the landscape boundary and there are no background edges to worry about. Similarly, on the **Aggregation** tab, if you select either the *Proximity index*, *Similarity index*, or *Connectance index* then you also need to specify additional information. These metrics are "functional" metrics and thus require additional parameterization. All three of these metrics require a search radius; the *Similarity index* also requires a similarity weights table (see above). To specify a search radius, click on the [...] button and enter the desired search radius in meters; e.g., 500. Note, a single search radius is specified for the *Proximity index* and *Similarity index*, and a separate threshold distance is specified for the *Connectance index*.

Lastly, click on the **Landscape metrics** button and then on each tabbed set of metrics. Again, you can choose a subset of metrics or simply "Select all" -- your choice. Note, on the **Diversity** tab, if you select *Relative Patch Richness*, then you also need to specify the maximum number of classes (or patch types). Simply click on the [...] button and enter the value; 6 in this case.

## 6. Specify additional parameters for the analysis

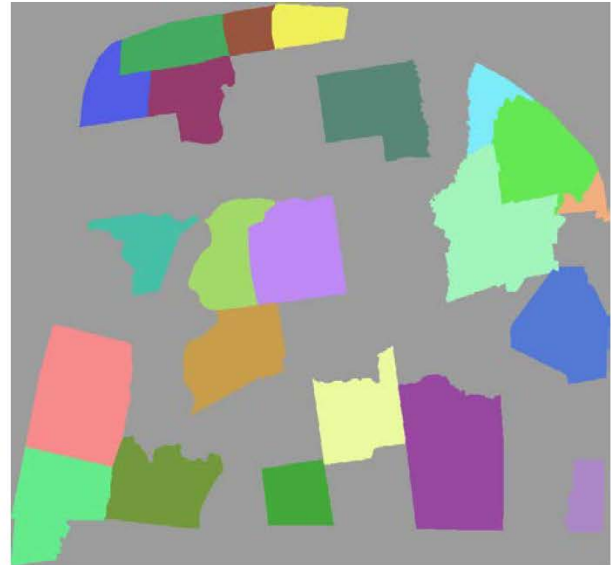
Next, you need to specify some additional parameters for the analysis. Click on the **Analysis parameters** tab on the left pane of the user interface. Here, is where you choose the neighbor rule for delineating patches (4 cell rule or 8 cell rule) and specify whether you want to sample the landscape to analyze sub-landscapes and, if so, by which method.

For this tutorial, keep the default 8 cell neighbor rule. With regards to sampling method, let's go through each method in turn to learn about what it is doing.

### 6.1 User-provided tiles

In this method, the landscape is subdivided into a set of mutually-exclusive and typically all-inclusive user-defined tiles (sub-landscapes). The tiles should not extend beyond the landscape boundary; in other words, the tiles should comprise the landscape of interest (i.e., the extent of positively valued cells). Moreover, the tile grid must have the same input data format and identical cell size and geographical alignment as the input landscape.

In our example, as shown below (left figure), the input landscape (lugrid) is subdivided into 40 tiles (sub-landscapes) representing townships. Each tile (town) has a unique integer-valued id ranging in value from 150 to 420 (note, the id's need not be consecutive). Each tile will be analyzed separately as a sub-landscape. However, FRAGSTATS will include a 1 cell wide border around each tile in which the cells are assigned negative their class value, designating that they are outside the landscape of



interest, but providing information on patch type adjacency for cells along the landscape boundary that will affect the edge-related metrics. Note, the tiles are mutually-exclusive and all-inclusive, and do not extent outside the landscape of interest (i.e., they do not extend into the nodata portion of the grid).

Note, it is not required that the tiles be all-inclusive, as in our example. For example, we could have a tiling scheme that allocates a small to large portion of the landscape to either background or nodata, as shown in the right figure above. In this particular case, it would not matter whether the unallocated portion of the landscape is designated as background or nodata, since both will be treated as negative background (external to the landscape of interest) by FRAGSTATS.

Select the **User-provided tiles** sampling option in the **Analysis parameters** tab on the left pane of the user interface, as shown here.

☒ User provided tiles

☐ Patch metrics

☒ Class metrics

☒ Landscape metrics

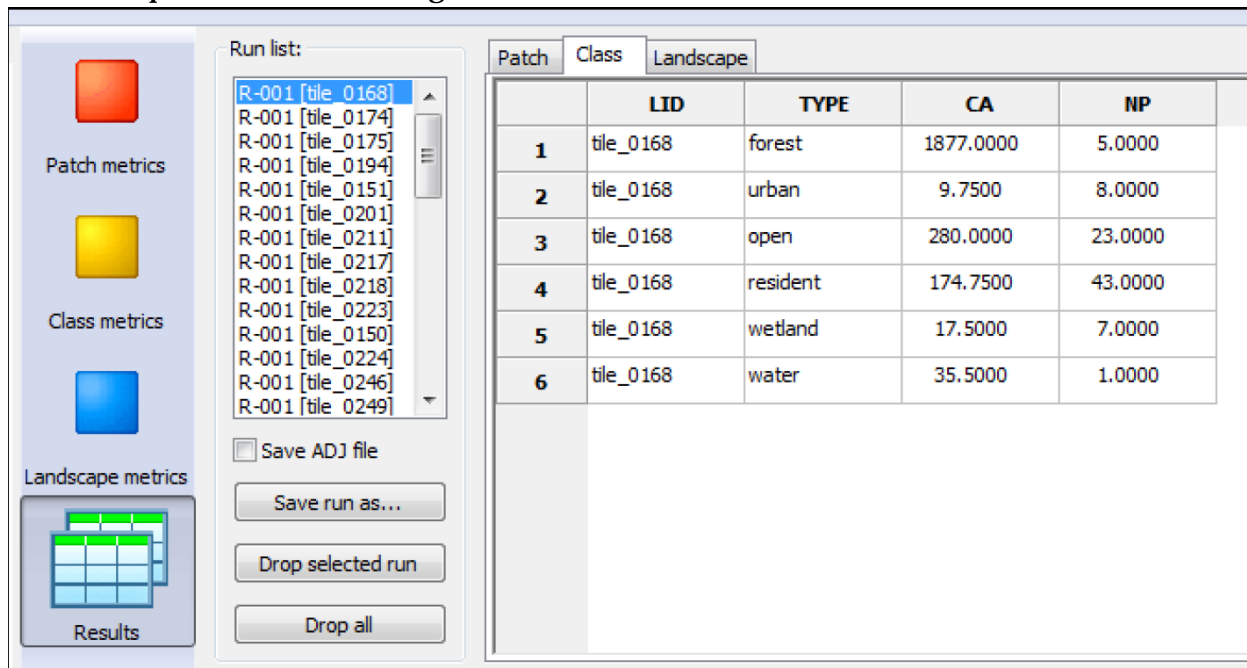
Tile grid:  ...

In addition, check the boxes for **Class** and **Landscape** metrics, as shown. Note, you must have at least one of the these boxes checked or you will get an error message later when trying to run the model. However, only check the level corresponding to the metrics you want to compute.

Next, import the tile grid. Simply click on the [...] button and repeat the process for inputting a grid, making sure that the data type is the same as before.

Next, you are ready to **Run** the model, as before (see tutorial #2). Simply click on the **Run** button, verify that the run parameters are correct and click on **Proceed**.

Lastly, once the run is complete, you can view the results, as before (see tutorial #2). The major difference between this run and the runs from the previous tutorials is that the **Run list** in the top-right pane of the user interface is going to contain a list of results pertaining to the tiles or sub-landscapes. In this case, the run list should contain 40 rows, one for each tile. Note, the LID field lists the tile number and this corresponds to the unique tile id in the tile grid.



The screenshot shows the FRAGSTATS user interface. On the left, there are four buttons: 'Patch metrics' (red), 'Class metrics' (yellow), 'Landscape metrics' (blue), and 'Results' (green). The 'Run list' pane in the center shows a list of 40 runs, with 'R-001 [tile\_0168]' selected. Below the list are buttons for 'Save ADJ file', 'Save run as...', 'Drop selected run', and 'Drop all'. On the right, the 'Landscape' tab is active, displaying a table with 5 columns: LID, TYPE, CA, and NP. The table contains 6 rows of data for tile\_0168.

	LID	TYPE	CA	NP
1	tile_0168	forest	1877.0000	5.0000
2	tile_0168	urban	9.7500	8.0000
3	tile_0168	open	280.0000	23.0000
4	tile_0168	resident	174.7500	43.0000
5	tile_0168	wetland	17.5000	7.0000
6	tile_0168	water	35.5000	1.0000

## 6.2 Uniform tiles

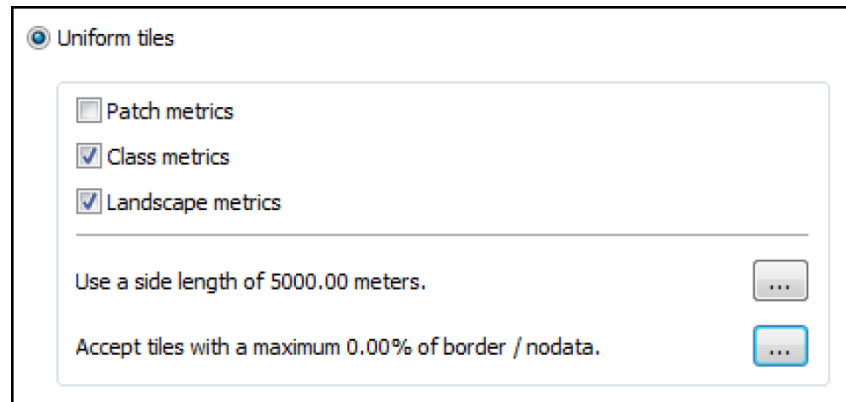
In this method, the landscape is subdivided into a set of mutually-exclusive and all-inclusive uniform square tiles (sub-landscapes) of a user-specified size. Note, in the current version of FRAGSTATS the uniform tiles are limited to squares, but we will eventually incorporate an option for hexagons.

With this option, FRAGSTATS will create a uniform grid of tiles of the size you specify that fills out the rectangular input grid starting from the top left corner. This means that if the landscape of interest is not rectangular, as in our example, that some of the uniform tiles will overlap the nodata portion of the input grid. Any tile that falls entirely within the nodata region of the input grid will be discarded by FRAGSTATS. However, any tile that partially overlaps the landscape of interest (i.e., positively valued cells in the

input grid) will be included or excluded depending on the user-specified preference for the maximum percentage of border/nodata to allow, as described below. In addition, FRAGSTATS automatically includes a 1 cell wide border around each tile in which the cells are assigned negative their class value, designating that they are outside the landscape of interest, but providing information on patch type adjacency for cells along the landscape boundary that will affect the edge-related metrics.

Let's see how this works. To begin, select the **Uniform tiles** sampling option in the **Analysis parameters** tab on the left pane of the user interface, as shown here.

In addition, check the boxes for **Class** and **Landscape** metrics, as shown. Note, you must have at least one of these boxes checked or you will get an error message later when trying to run the model. However, only check the level corresponding to the metrics you want to compute.



Next, specify the size of the square tile to use in meters. Simply click on the [...] button and enter the side length in meters. Let's enter **5000** m (5 km) for this example.

Next, you have the option of accepting tiles with a maximum user-specified percentage of border/nodata. The default is **0%**, which means that any tile that contains even 1 cell of either border (negative cells) or nodata will be discarded. Let's keep the default for now and see what happens.

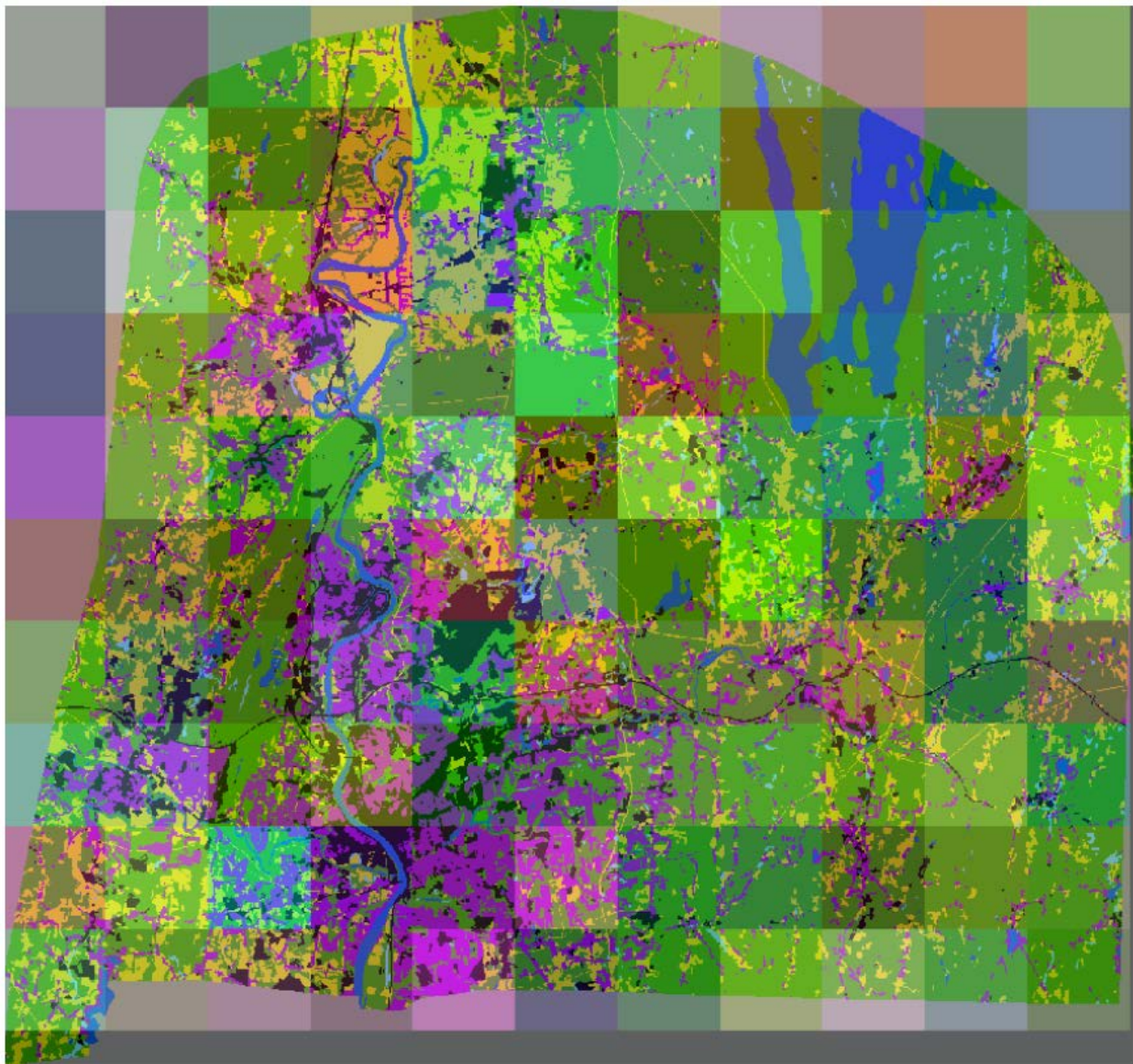
Next, you are ready to **Run** the model, as before (see tutorial #2). Simply click on the **Run** button, verify that the run parameters are correct and click on **Proceed**.

Lastly, once the run is complete, you can view the results, as before (see tutorial #2). The major difference between this run and the runs from the previous tutorials is that the **Run list** in the top-right pane of the user interface is going to contain a list of results pertaining to the uniform tiles or sub-landscapes. In this case (but not shown here), the run list should contain 66 rows, one for each valid tile. The SUMMARY in the Activity log should indicate that there were a total of 110 tiles, but that only 66 were deemed valid based on the threshold of 0% border/nodata. Note, the LID field lists the tile number and this corresponds to the unique tile id in the tile grid that is output by FRAGSTATS.



Let's view the uniform tile grid and evaluate its correspondence with the FRAGSTATS results. Here, I will use ESRI ArcMap, but if you are not working with ESRI ArcGIS, and you analyzed the provided ascii grid (lugrid.asc), you can use R to view the ascii tile grid that would have been created using the procedures outlined in tutorial #1, but modifying the script accordingly for the created ascii grid (**tiles00001**).

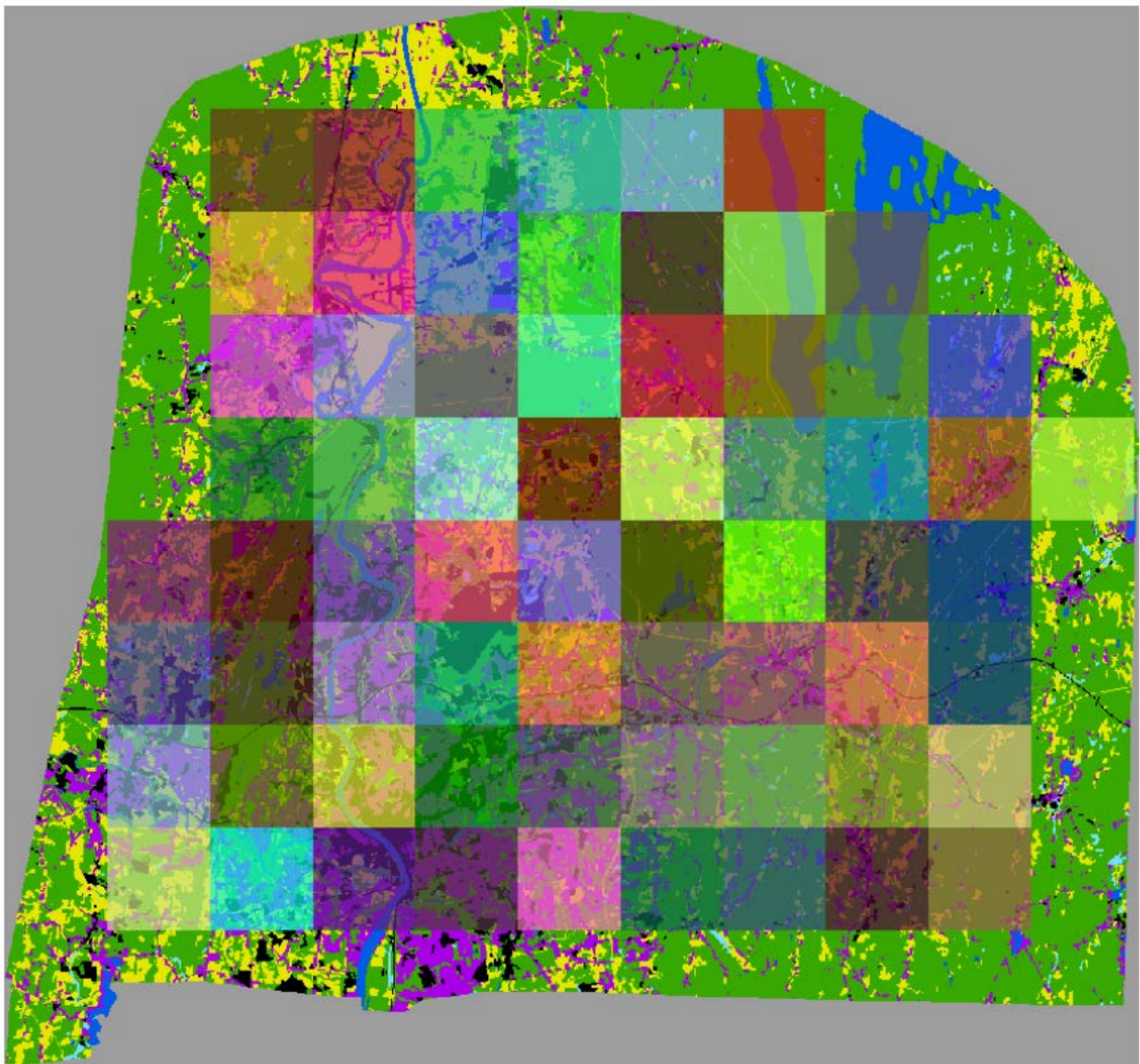
Open up the provided **fragtutorial\_6.mxd** project in ArcMap, if it is not already open from earlier. Add the created uniform tile grid (**tiles00001.tif**) to the project. Note, each time FRAGSTATS creates a tile grid in the same directory, it will increment the tile grid name by 1. Since this is presumably your first run in this directory, the created tile grid should be numbered 00001.



For our purposes, I changed the symbology to give a different color to each unique tile and made them 60% transparent. There are two important things to note about the tile grid:

1. Some of the tiles fall entirely within the nodata portion of the input landscape (lugrid), while others partially or completely overlap the landscape of interest.
2. There is a strip of nodata at the bottom and on the right side of the tile grid because the grid dimensions were not perfectly divisible by the user-specified tile size (5,000 m in this case) and the tiling begins in the top left.

Importantly, because we specified a maximum threshold of 0% border/nodata, any tile containing even a single cell of nodata was discarded. In fact, the result summary indicates that 44 tiles were discarded out of the 110 total tiles, leaving 66 valid tiles. In

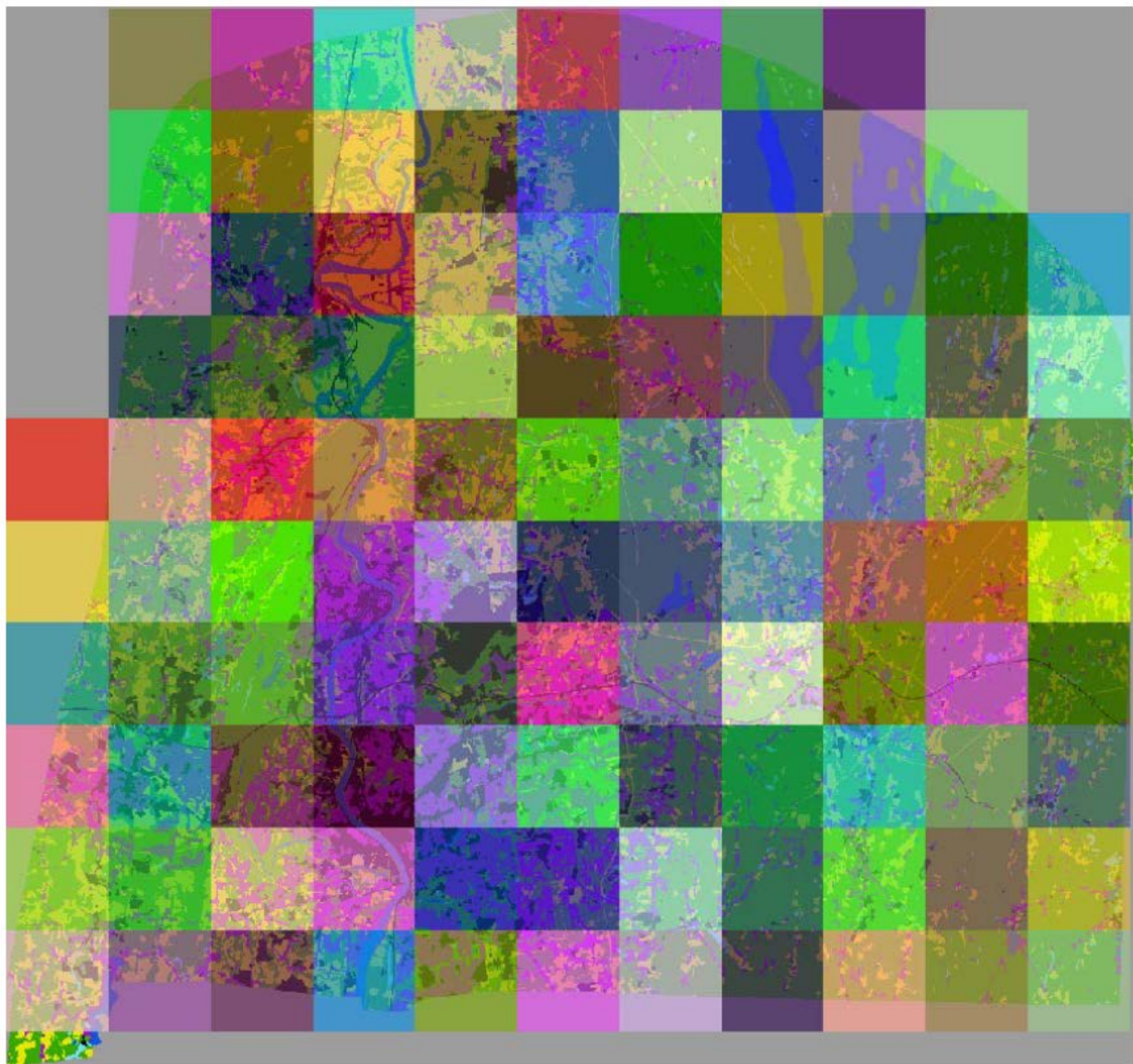




this figure, only the valid tiles are shown on top of the input grid.

Now, let's change the maximum percentage of border/nodata from 0% to say 100% and see what happens. Note, a 100% doesn't actually mean that a tile that is composed entirely of nodata will be deemed valid, since this would be nonsensical. A valid tile still has to have at least one cell in the landscape of interest, regardless. However, this means that any tile intersecting at least 1 cell of the landscape will be deemed valid.

Change the threshold to **100%** and run the analysis and review the results. The run list (not shown here) should contain 103 rows, one for each valid tile. The SUMMARY in the Activity log should indicate that there were a total of 110 tiles, and that 103 were deemed valid. Here's the set of tiles that were analyzed in this scenario. Again, the valid tiles are shown on top of the input grid.





## 6.3 User-provided points

In this method, the user provides a set of points (in a formatted table) or focal cells (in a grid) to serve as the center of windows (sub-landscapes) of a user-specified size and shape. Any point that falls within the nodata region of the input grid will be summarily discarded by FRAGSTATS. Any point that falls within the landscape of interest (i.e., positively valued cells in the input grid) will be included or excluded depending on the user-specified preference for the maximum percentage of border/nodata in the window to allow, as described below. In addition, FRAGSTATS automatically includes a 1 cell wide border around each window in which the cells are assigned negative their class value, designating that they are outside the landscape of interest, but providing information on patch type adjacency for cells along the landscape boundary that will affect the edge-related metrics.

Let's see how this works. To begin, select the **User-provided points** sampling option in the **Analysis parameters** tab on the left pane of the user interface, as shown here.

In addition, check the boxes for **Class** and **Landscape** metrics, as shown. Note, you must have at least one of the these boxes checked or you will get an error message later when trying to run the model. However, only check the level corresponding to the metrics you want to compute.

The screenshot shows the 'User provided points' dialog box. It has a title bar and a main area with several sections. At the top, the 'User provided points' radio button is selected. Below this, there are three checkboxes for metrics: 'Patch metrics' (unchecked), 'Class metrics' (checked), and 'Landscape metrics' (checked). A horizontal line separates this from the next section. In the next section, there are two radio buttons for window shape: 'Round with a radius of 5000.00 meters.' (selected) and 'Square with a side length of 100.00 meters.' (unchecked). Below these are three text input fields with '...' buttons: 'Accept tiles with a maximum 0.00% of border / nodata.' (selected), and two others for file paths. The first file path is 'C:\Work\Fragstats\Tutorial\Tutorial\_6\points' and the second is 'points.fpt'. The 'Points grid' radio button is selected, and the 'Points table' radio button is unchecked.

Next, specify the shape (round or square) and size (in meters) of the window to use. Simply click on the [...] button and enter the radius (for round) or side length (for square) in meters. Let's choose a **round** window and enter **5000** m (5 km) for this example.

Next, you have the option of accepting tiles with a maximum user-specified percentage of border/nodata. The default is **0%**, which means that any window that contains even 1 cell of either border (negative cells) or nodata will be discarded. Let's keep the default for now and see what happens.

Next, you have the option of reading in a points grid or points table to identify the focal cells. The points grid must have the same input data format and identical cell size and

geographical alignment as the input landscape. The grid should contain a unique non-zero integer value for each focal cell (point) of interest; all others should be set to nodata. The points table must have the following format.

FPT\_TABLE

[first point id#: first point row#: first point col#]

[second point id#: second point row#: second point col#]

etc.

Note, each bracketed item contains point coordinates of the following form: [id : row : column] or [id:row:column], where point id values must be unique integer values (duplicates will be ignored), row and column values must be integer values within the ranges specific to the target dataset, and represent row and column numbers not geographic coordinates (out-of-range and duplicate coordinates will be ignored). For example, the first few lines of the points table provided for this example looks like this:

FPT\_TABLE

[1:968:1002]

[2:968:935]

[3:965:63]

etc.

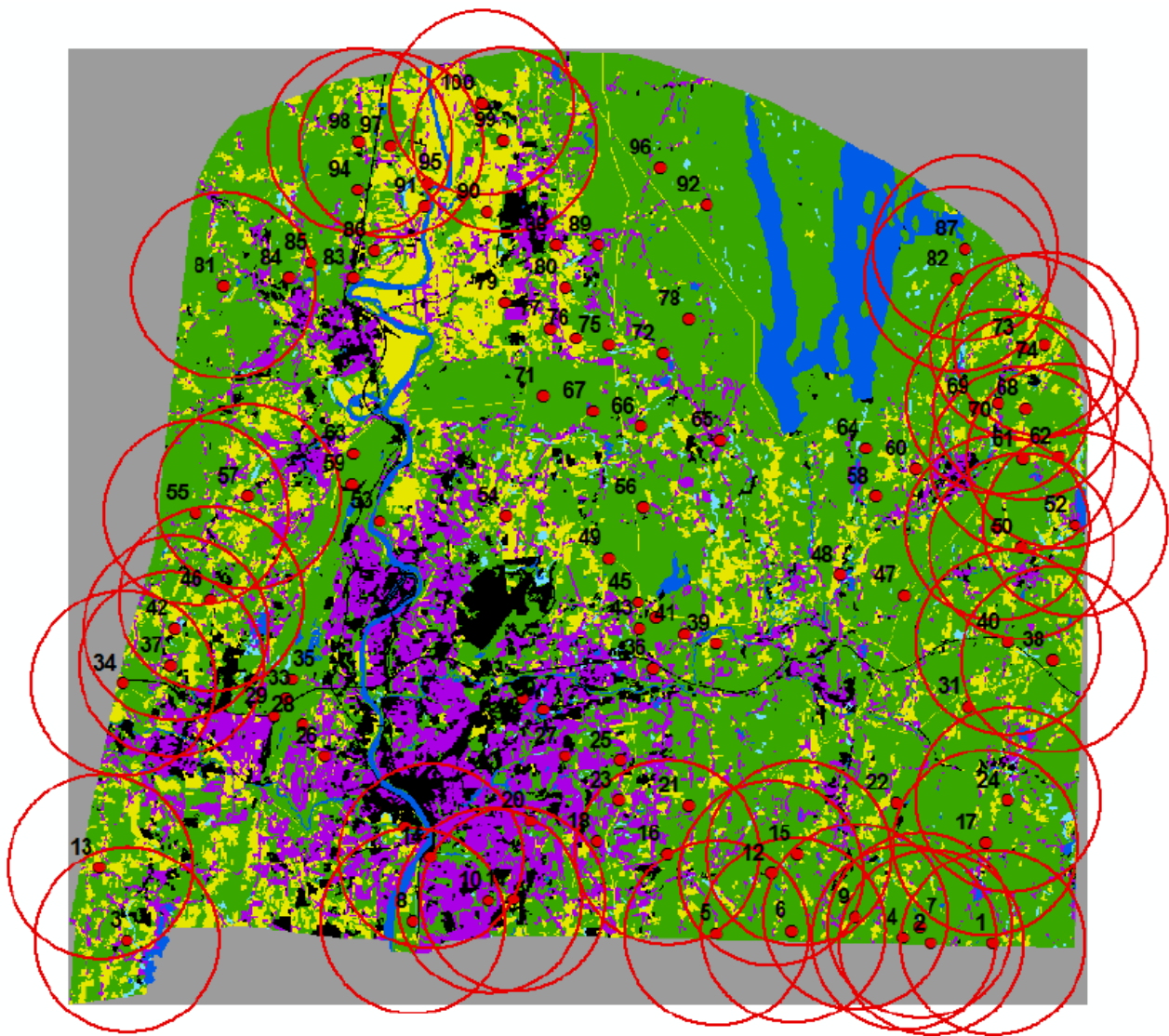
Choose either the points grid or points table to load by clicking on the corresponding radio button. To import the points grid (**points**), simply click on the [...] button and repeat the process for inputting a grid, making sure that the data type is the same as before. To import the points table, simply click on the [...] button, navigate to the tutorial folder, and select the **points.fpt** file. In both cases, there are 100 points or focal cells identified.

Next, you are ready to **Run** the model, as before (see tutorial #2). Simply click on the **Run** button, verify that the run parameters are correct and click on **Proceed**.

Lastly, once the run is complete, you can view the results, as before (see tutorial #2). The major difference between this run and the runs from the previous tutorials is that the **Run list** in the top-right pane of the user interface is going to contain a list of results pertaining to the uniform tiles or sub-landscapes. In this case (but not shown here), the run list should contain 58 rows, one for each valid window. The SUMMARY in the Activity log should indicate that there were a total of 74 windows considered (out of 100 points), but that only 58 of these were deemed valid based on the threshold of 0% border/nodata and 16 were skipped because their windows included one or more border/nodata cells. The remaining 26 points were never even considered because their windows extended beyond the edge of the rectangular grid. Note, the LID field lists the point number and this corresponds to the unique point id in the points grid/table.

Let's view the points grid and evaluate its correspondence with the FRAGSTATS results. Here, I will use ESRI ArcMap, but if you are not working with ESRI ArcGIS, and you analyzed the provided ascii grid (lugrid.asc), you can use R to view the ascii points grid that would have been created using the procedures outlined in tutorial #1, but modifying the script accordingly for the name of the ascii grid (**points**).

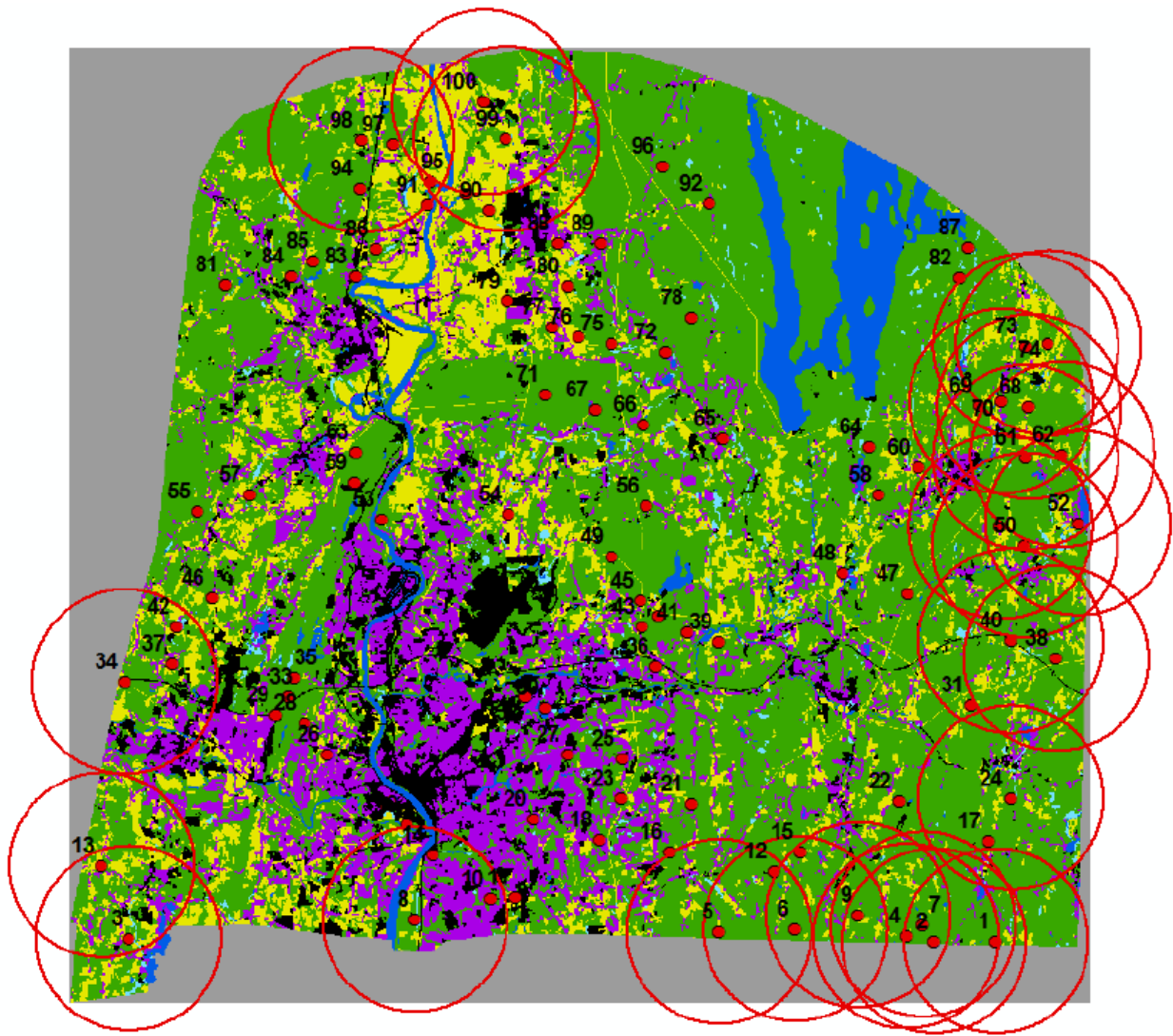
Open up the provided **fragtutorial\_6.mxd** project in ArcMap, if it is not already open from earlier. Note, the ArcMap project contains the points grid (**points.tif**), as well as a points shapefile (**points shapefile**) included for the sole purpose of facilitating the display of the points. As shown here, the 100 points (red dots) are all located inside the landscape of interest (i.e., none fall in the nodata region), but vary in their distance to the landscape boundary (i.e., the edge of the landscape of interest) and the edge of the rectangular grid.



In fact, 26 points are within 5 km of the edge of the rectangular input grid and were summarily discarded by FRAGSTATS. These points are indicated in the figure below by having circular buffers that extend beyond the edge of the grid. An additional 16 points are within 5 km of the landscape boundary (i.e., the edge of the positively valued cells and the landscape of interest) and were discarded because they did not meet the 0% border/nodata threshold. This leaves 58 valid windows for the analysis.

Now, let's change the maximum percentage of border/nodata from 0% to say 100% and see what happens. As noted above, a 100% doesn't actually mean that a window that is composed entirely of nodata will be deemed valid. A valid window still has to have at least one cell in the landscape of interest, regardless. However, this means that any window intersecting at least 1 cell of the landscape will be deemed valid.

Change the threshold to **100%** and run the analysis and review the results. The run list (not shown here) should contain 74 rows, one for each valid window. The SUMMARY in the Activity log should indicate that there were a total of 74 windows, and that 74 were deemed valid. This is because the only windows that were summarily discarded were the 26 that extend beyond the edge of the rectangular input grid. All the other windows fall within the rectangular grid and contain at least 1 cell that is not border/nodata. The image below shows the 26 points that were discarded.



#### ***6.4 Random points without overlap***

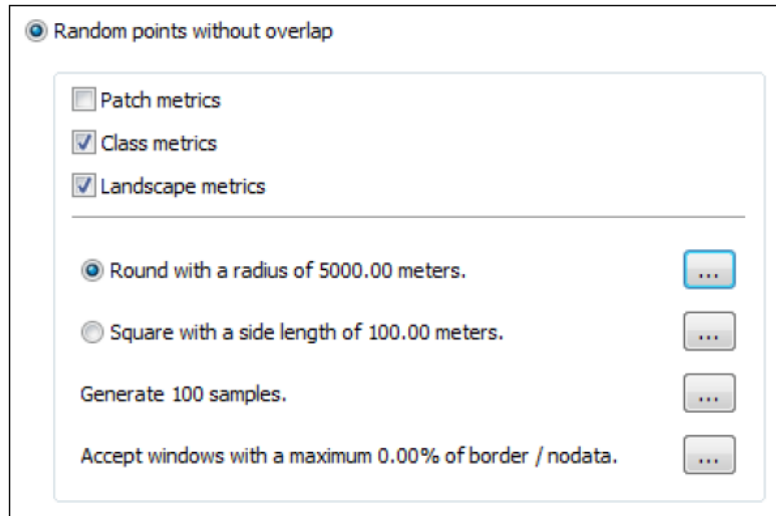
In this method, FRAGSTATS generates random point locations to serve as the center of windows (sub-landscapes) of a user-specified size and shape such that the windows do not overlap. The random points are always greater than or equal to the radius of the window from the edge of the rectangular input grid, and thus none are summarily discarded as can happen with user-provided points. However, within this constraint, the random distance from the edge of the landscape of interest (i.e., positively valued cells in the input grid) depends on the user-specified preference for the maximum percentage of border/nodata in the window to allow, as described below. In addition, FRAGSTATS automatically includes a 1 cell wide border around each window in which the cells are assigned negative their class value, designating that they are outside the landscape of



interest, but providing information on patch type adjacency for cells along the landscape boundary that will affect the edge-related metrics.

Let's see how this works. To begin, select the **Random points without overlap** sampling option in the **Analysis parameters** tab on the left pane of the user interface, as shown here.

In addition, check the boxes for **Class** and **Landscape** metrics, as shown. Note, you must have at least one of these boxes checked or you will get an error message later when trying to run the model. However, only check the level corresponding to the metrics you want to compute.



The screenshot shows a dialog box titled "Random points without overlap". It contains several options and input fields:

- ☐ Patch metrics
- ☒ Class metrics
- ☒ Landscape metrics
- ☒ Round with a radius of 5000.00 meters. (with a [...] button)
- ☐ Square with a side length of 100.00 meters. (with a [...] button)
- Generate 100 samples. (with a [...] button)
- Accept windows with a maximum 0.00% of border / nodata. (with a [...] button)

Next, specify the shape (round or square) and size (in meters) of the window to use. Simply click on the [...] button and enter the radius (for round) or side length (for square) in meters. Let's choose a **round** window and enter **5000** m (5 km) for this example.

Next, specify the number of random samples (or point locations) to use; the default is **100**. Simply click on the [...] button and enter the sample size. Let's keep the default for now and see what happens.

Next, you have the option of accepting tiles with a maximum user-specified percentage of border/nodata. The default is **0%**, which means that FRAGSTATS will not generate a random window that contains even 1 cell of either border (negative cells) or nodata. Let's keep the default for now and see what happens.

Next, you are ready to **Run** the model, as before (see tutorial #2). Simply click on the **Run** button, verify that the run parameters are correct and click on **Proceed**.

Lastly, once the run is complete, you can view the results, as before (see tutorial #2). The major difference between this run and the runs from the previous tutorials is that the **Run list** in the top-right pane of the user interface is going to contain a list of results pertaining to the random windows or sub-landscapes. In this case (but not shown here), the run list should contain multiple rows, one for each randomly generated window. The SUMMARY in the Activity log should indicate that there were a total of somewhere around 14 random windows generated (out of a maximum desired 100).

FRAGSTATS attempts to reach the user-specified sample size, but if it fails to create a valid window after 5,000 attempts it stops. The issue here is that windows cannot overlap with this sampling option and a 5 km window is pretty big for this landscape, especially considering that the windows cannot include even a single cell of border/nodata. Around 14 random windows are all that can fit given these constraints. Note, the LID field lists the point number and this corresponds to the unique point id in the generated points grid.

Let's view the FRAGSTATS generated points grid and evaluate its correspondence with the FRAGSTATS results. Here, I will use ESRI ArcMap, but if you are not working with ESRI ArcGIS you can use your GIS and the appropriate comparable methods to view the generated points grid, or if you analyzed the provided ascii grid (lugrid.asc) you can use R to view the ascii points grid that would have been created using the procedures outlined in tutorial #1, but modifying the script accordingly for the name of the generated ascii grid (**points00001.asc**).

**Working with ArcMap:** Open up the provided **fragtutorial\_6.mxd** project in ArcMap, if it is not already open from earlier. Note, the ArcMap project contains the points grid (**points.tiff**), as well as a points shape file (**points shape**) included for the sole purpose of facilitating the display of the points, but these are the user-provided points that we worked with earlier. You need to add the FRAGSTATS generated random points grid that was just created (e.g., **points00001.tif**). Note, the random points are extremely difficult to see in their grid form because the focal cells representing the points can't be displayed larger than they are, so you have to zoom in to see them. You will want to convert the grid to a shapefile in order to be able to enlarge the points for display and then you can buffer the points with a 5,000 m circular buffer to see what happened. Here's how you can do that:

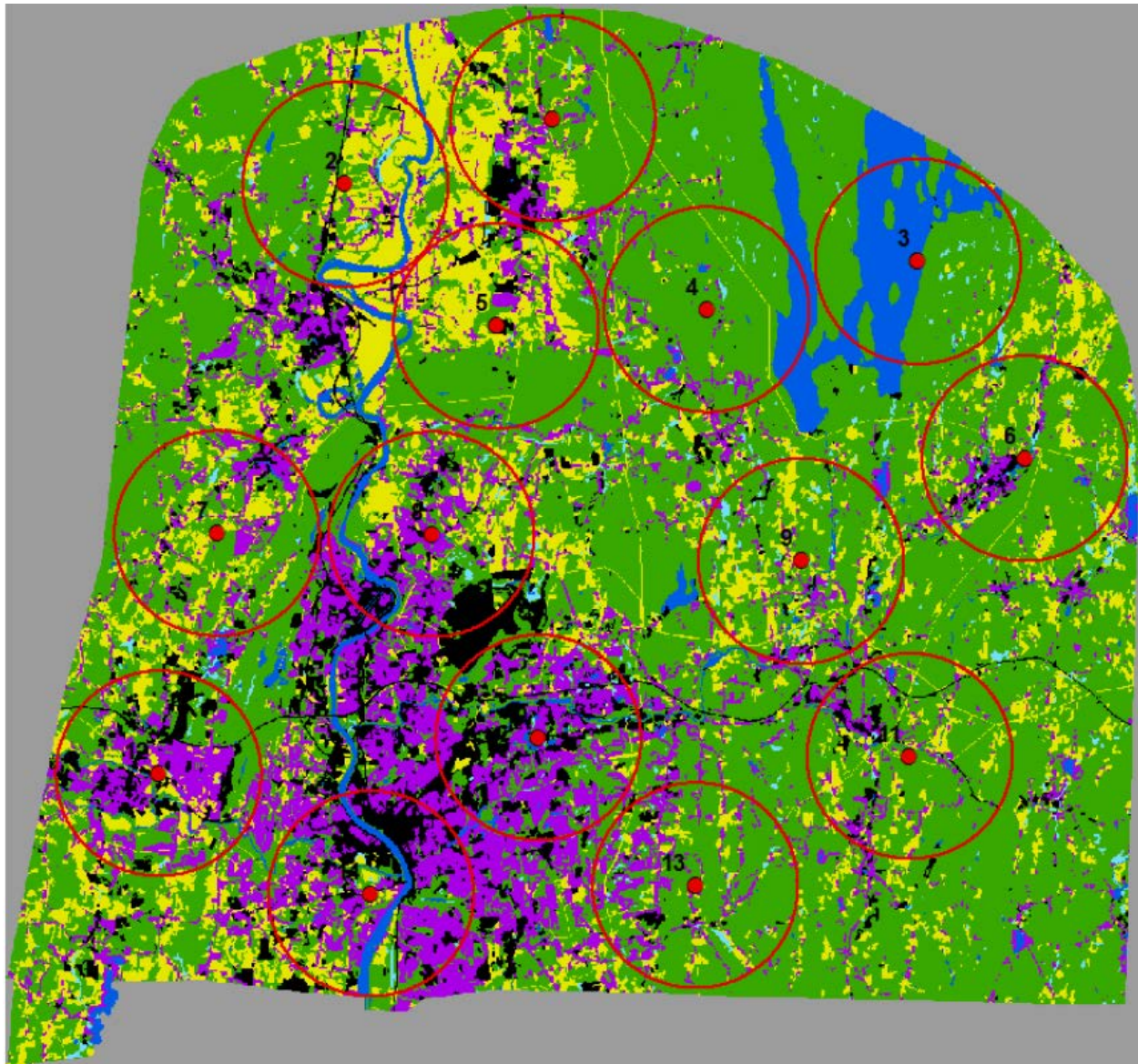
1. Open the Arc Toolbox and select "Conversion Tool", then "From Raster", and then "Raster to Point". Use the dropdown button to select **points00001.tif** as the Input Raster, choose "Value" as the Field, and enter a path and file name (e.g., **points00001**) for the Output File. Click on OK at the bottom of the window and keep your fingers crossed that it doesn't crash.
2. Add the newly created shapefile (e.g., **points00001**) to the table of contents. Note, this shapefile has a point for every cell in the original grid and so we need to select just the actual points of interest.
3. From the Arc Toolbox select "Analysis Tools", then "Extract", and then "Select". Use the dropdown button to select the newly created shapefile (e.g., **points00001**) as the Input Features, enter a path and file name (e.g., **points00001x**) and enter the following Expression: "GRID\_CODE" >0. Click

on OK at the bottom of the window and keep your fingers crossed that it doesn't crash.

4. Add the newly created shapefile (e.g., **points00001x**) to the table of contents. Note, this shapefile has a point for each of the grid cells with a value >0, which in this case is going to be around 14 or so depending on the random placement of non-overlapping circles. You can modify the symbology of the points to expand the size of the points so that you can see them better.
5. Next, if you want to add the x-y coordinates of the points to the attribute table of the shapefile so that if you choose to export the data you have the geographic coordinates of the points, from the Arc Toolbox select "Data Management Tools", then select "Features", and then select "Add XY Coordinates". Select the newly created shapefile (e.g., **points00001x**) for the Input Features and click on OK at the bottom of the window.
6. Lastly, you can add a buffer to the points to see what the specified window around each point looks like, in this case a 5 km radius window. From the Arc Toolbox select "Analysis Tools", then select "Proximity", and then select "Buffer". Select the newly created shapefile (e.g., **points00001x**) for the Input Features, enter a path and file name (e.g., **pointsbuff**) for the "Output Feature Class", enter 5000 for the "Linear unit" and make sure the units are set to "meters", and click on OK at the bottom of the window. With any luck the buffers should appear after a while and you can change the symbology to suit.

As shown below, in my particular run, 14 random points (windows) were generated out of the maximum desired 100 points. Note, the 5 km windows are all contained entirely within the landscape of interest (i.e., none of them include even a single cell of border/nodata because we specified a 0% threshold) and are mutually exclusive (i.e., do not overlap because we specified random points 'without' overlap).



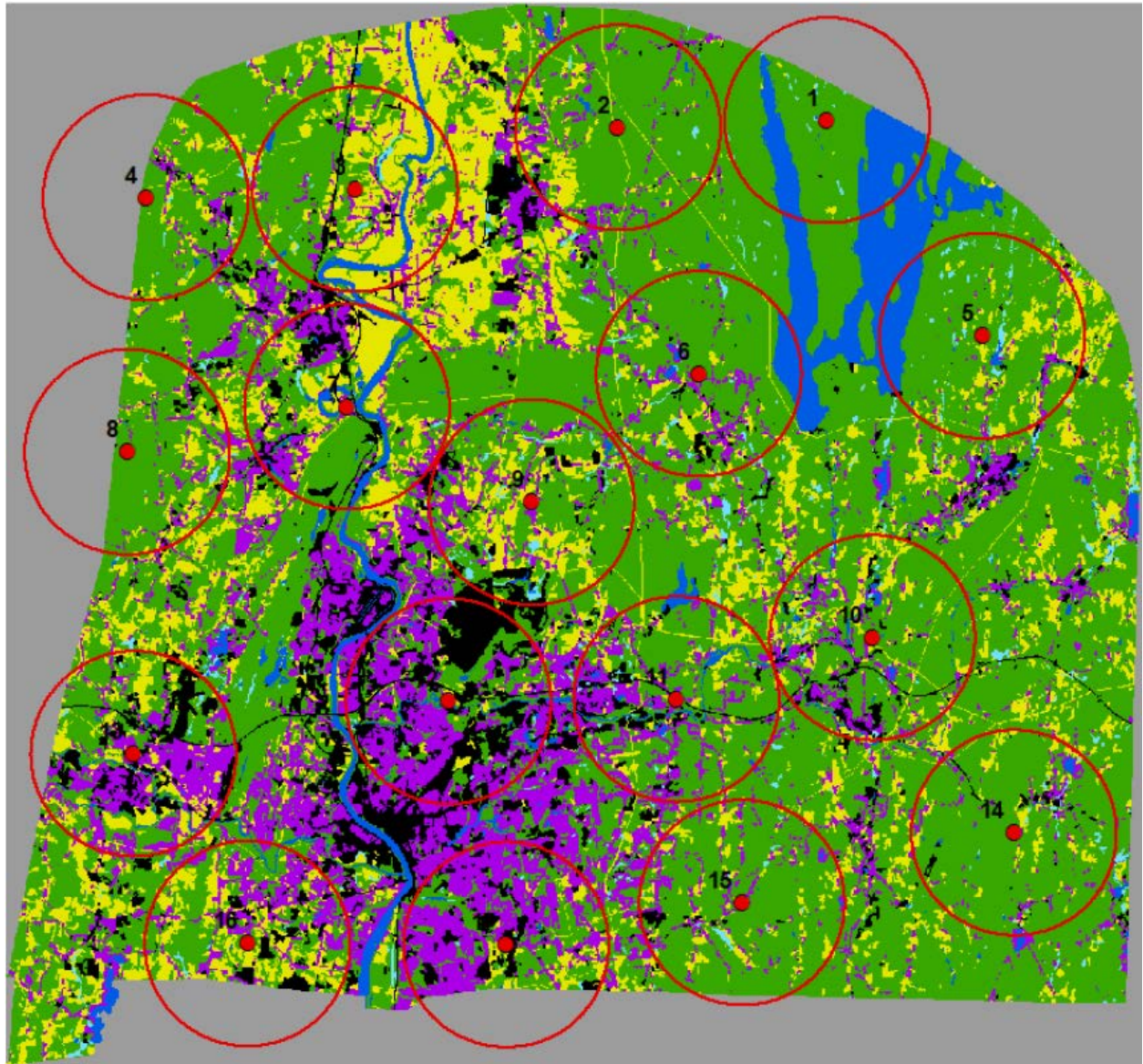


Now, let's change the maximum percentage of border/nodata from 0% to say 100% and see what happens. As noted above, a 100% doesn't actually mean that a window that is composed entirely of nodata will be deemed valid. A valid window still has to have at least one cell in the landscape of interest, regardless. However, this means that any window intersecting at least 1 cell of the landscape will be deemed valid.

Change the threshold **100%** and run the analysis and review the results. The run list (not shown here) should contain multiple rows, one for each randomly generated window. The SUMMARY in the Activity log should indicate that there were a total of somewhere around 17 random windows generated (out of a maximum desired 100). The image below shows the result for my particular run, in which FRAGSTATS generated 17 random windows without overlap. The main difference between this run and the previous run is that the random windows are allowed to be closer to the edge of the



landscape of interest because they can contain any percentage of border/nodata, so long as the focal cells still fall within the landscape of interest and the windows do not extend beyond the edge of the rectangular input grid.



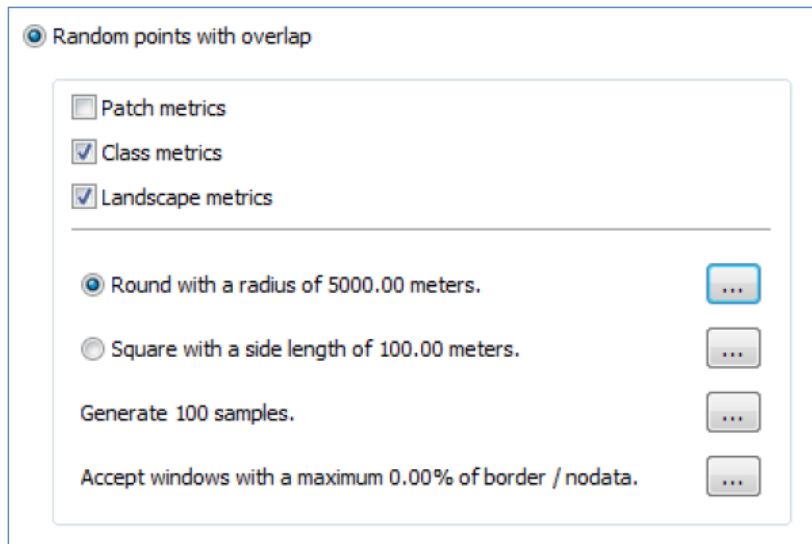
### ***6.5 Random points with overlap***

In this method, FRAGSTATS generates random point locations to serve as the center of windows (sub-landscapes) of a user-specified size and shape, but allows the windows to overlap. The random points are always greater than or equal to the radius of the window from the edge of the rectangular input grid, and thus none are summarily discarded as can happen with user-provided points. However, within this constraint, the random distance from the edge of the landscape of interest (i.e., positively valued cells in the input grid) depends on the user-specified preference for the maximum percentage of

border/nodata in the window to allow, as described below. In addition, FRAGSTATS automatically includes a 1 cell wide border around each window in which the cells are assigned negative their class value, designating that they are outside the landscape of interest, but providing information on patch type adjacency for cells along the landscape boundary that will affect the edge-related metrics.

Let's see how this works. To begin, select the **Random points with overlap** sampling option in the **Analysis parameters** tab on the left pane of the user interface, as shown here.

In addition, check the boxes for **Class** and **Landscape** metrics, as shown. Note, you must have at least one of these boxes checked or you will get an error message later when trying to run the model. However, only check the level corresponding to the metrics you want to compute.

A screenshot of the 'Random points with overlap' sampling options in the FRAGSTATS software. The interface is titled 'Random points with overlap' and contains several settings. At the top, there are three checkboxes: 'Patch metrics' (unchecked), 'Class metrics' (checked), and 'Landscape metrics' (checked). Below these, there are two radio button options for the window shape: 'Round with a radius of 5000.00 meters.' (selected) and 'Square with a side length of 100.00 meters.' (unselected). To the right of each radio button is a button with three dots (...). Below the shape options, there is a text field for 'Generate 100 samples.' with a three-dot button to its right. At the bottom, there is a text field for 'Accept windows with a maximum 0.00% of border / nodata.' with a three-dot button to its right.

Next, specify the shape (round or square) and size (in meters) of the window to use. Simply click on the [...] button and enter the radius (for round) or side length (for square) in meters. Let's choose a **round** window and enter **5000 m** (5 km) for this example.

Next, specify the number of random samples (or point locations) to use; the default is **100**. Simply click on the [...] button and enter the sample size. Let's keep the default for now and see what happens.

Next, you have the option of accepting tiles with a maximum user-specified percentage of border/nodata. The default is **0%**, which means that FRAGSTATS will not generate a random window that contains even 1 cell of either border (negative cells) or nodata. Let's keep the default for now and see what happens.

Next, you are ready to **Run** the model, as before (see tutorial #2). Simply click on the **Run** button, verify that the run parameters are correct and click on **Proceed**.

Lastly, once the run is complete, you can view the results, as before (see tutorial #2). The major difference between this run and the runs from the previous tutorials is that the **Run list** in the top-right pane of the user interface is going to contain a list of

results pertaining to the random windows or sub-landscapes. In this case (but not shown here), the run list should contain multiple rows, one for each randomly generated window. The SUMMARY in the Activity log should indicate that there were a total of 100 windows generated. Note, in contrast to the random points without overlap sampling option, with the random points with overlap sampling option, FRAGSTATS will always generate the user-specified number of windows. Note, the LID field lists the point number and this corresponds to the unique point id in the generated points grid.

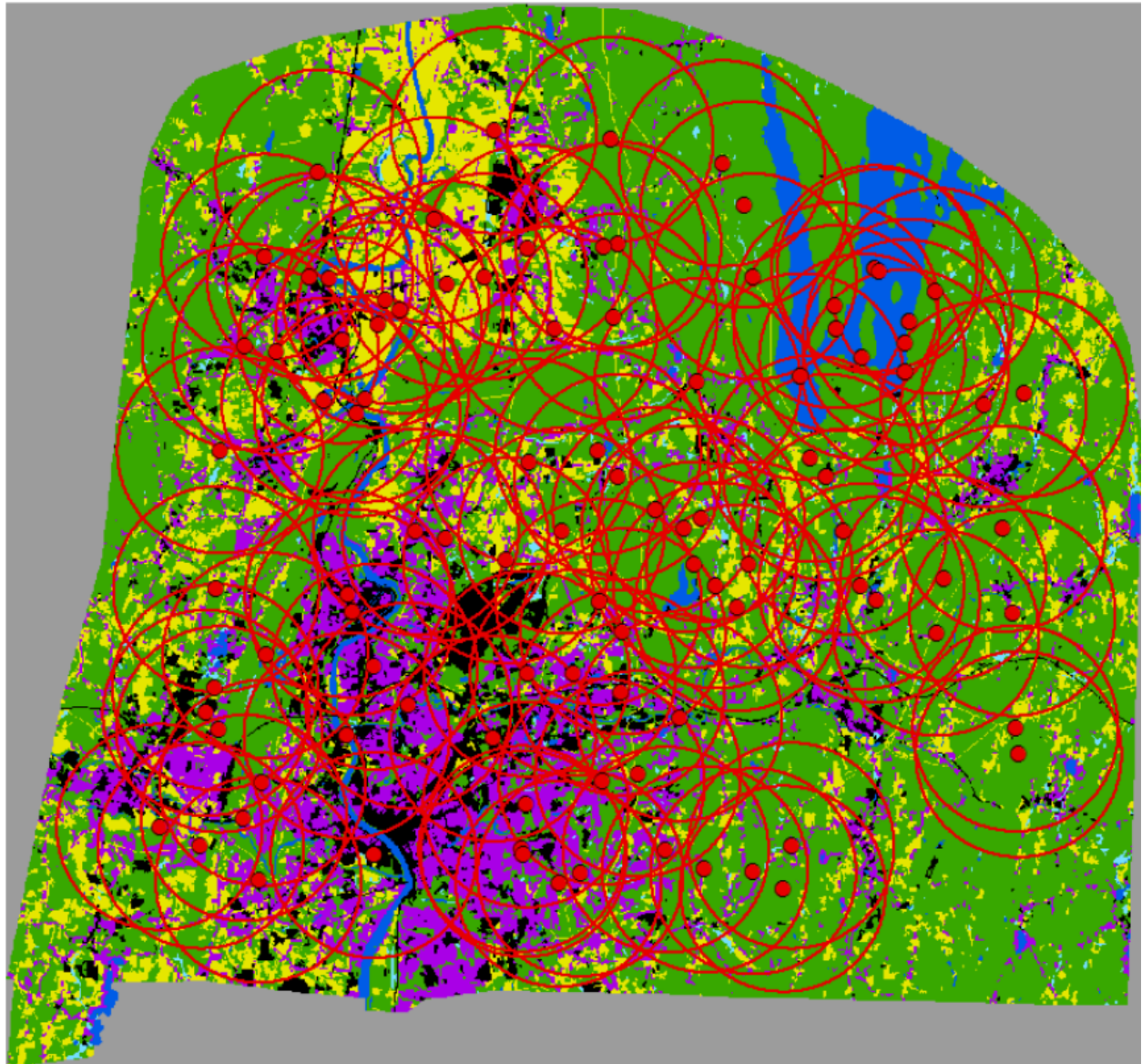
Let's view the FRAGSTATS generated points grid and evaluate its correspondence with the FRAGSTATS results. Here, I will use ESRI ArcMap, but if you are not working with ESRI ArcGIS you can use your GIS and the appropriate comparable methods to view the generated points grid, or you can use R to view the ascii points grid that would have been created using the procedures outlined in tutorial #1, but modifying the script accordingly for the name of the generated ascii grid (e.g., **points00002.asc**).

***Working with ArcMap:*** Open up the provided **fragtutorial\_6.mxd** project in ArcMap, if it is not already open from earlier. Note, the ArcMap project contains the points grid (**points.tif**), as well as a points shape file (**points shape**) included for the sole purpose of facilitating the display of the points, but these are the user-provided points that we worked with earlier. You need to add the FRAGSTATS generated random points grid that was just created (e.g., **points00002.tif**). Note, the random points are extremely difficult to see in their grid form because the focal cells representing the points can't be displayed larger than they are, so you have to zoom in to see them. You will want to covert the grid to a shapefile in order to be able to enlarge the points for display and then you can buffer the points with a 5,000 m circular buffer to see what happened. You can follow the procedures outlined above demonstrated for the random points without overlap.

As shown below for my particular run, FRAGSTATS generated 100 random points (windows), but allowing them to overlap. Note, the 5 km windows are all contained entirely within the landscape of interest (i.e., none of them include even a single cell of border/nodata because we specified a 0% threshold) and are overlapping.

Now, let's change the maximum percentage of border/nodata from 0% to say 100% and see what happens. As noted above, a 100% doesn't actually mean that a window that is composed entirely of nodata will be deemed valid. A valid window still has to have at least one cell in the landscape of interest, regardless. However, this means that any window intersecting at least 1 cell of the landscape will be deemed valid.





Change the threshold **100%** and run the analysis and review the results. The run list (not shown here) should contain multiple rows, one for each randomly generated window. The SUMMARY in the Activity log should indicate that there were a total of 100 windows generated. The image below shows the result for my particular run, in which FRAGSTATS generated 100 random windows, but allowing them to overlap. The main difference between this run and the previous run is that the random windows are allowed to be closer to the edge of the landscape of interest because they can contain any percentage of border/nodata, so long as the focal cells still fall within the landscape of interest and the windows do not extend beyond the edge of the rectangular input grid.

