Last Updated: 16 February 2023

Prepared by: Kevin McGarigal

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.

2. Create a FRAGSTATS model

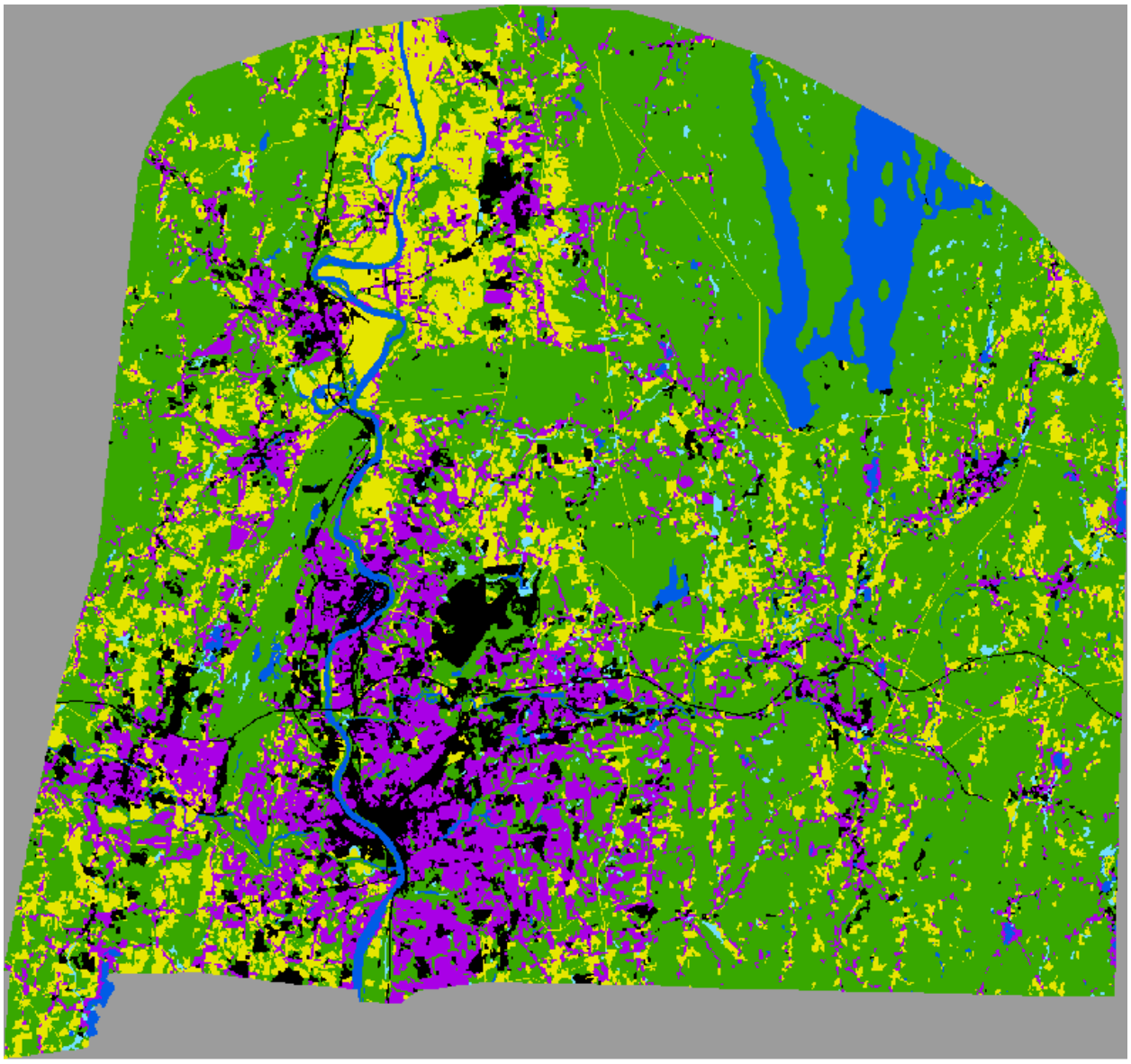
Next, create a New FRAGSTATS model, as before (see tutorial #2). Simply click on the **New** button on the tool bar or select **New** from the File drop-down menu. This creates a blank model for you to parameterize.

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, simply import the GeoTIFF; 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 ArcMap, 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 ArcMap, 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

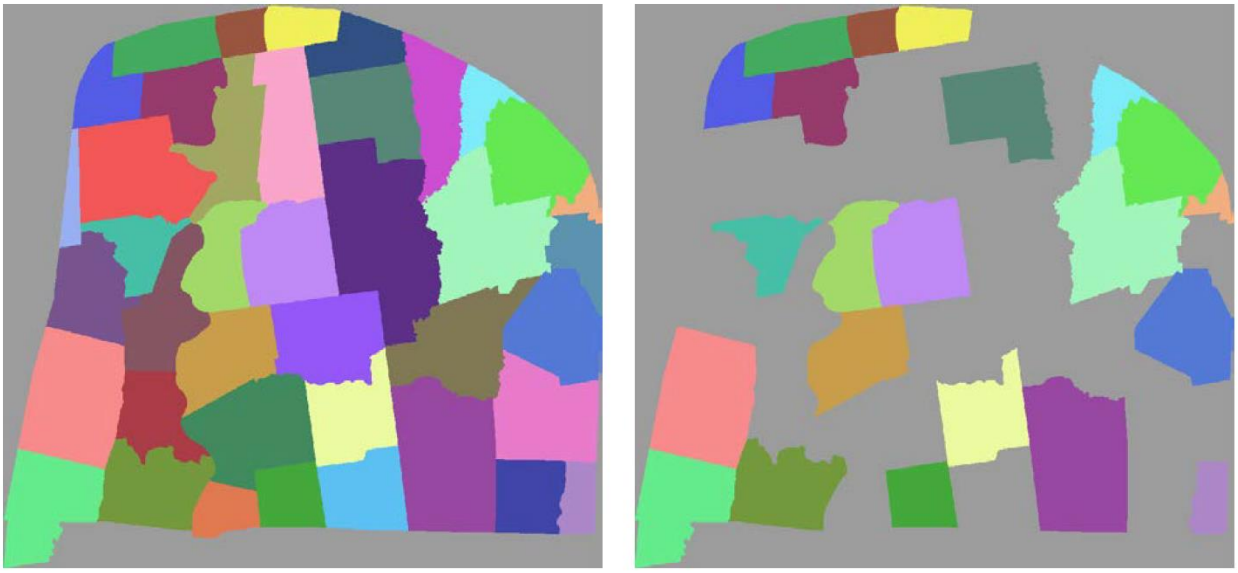
Next, you need to specify some additional parameters for the analysis. Click on the **Analysis parameters** tab on the left pane of the user interface. Here, is where you chose the neighbor rule for delineating patches (4 cell rule or 8 cell rule) and specify whether you want to sample the landscape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

For this tutorial, keep the default 8 cell neighbor rule. 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 grid extent (i.e., same number of rows and columns), cell size and cell alignment as the input landscape.

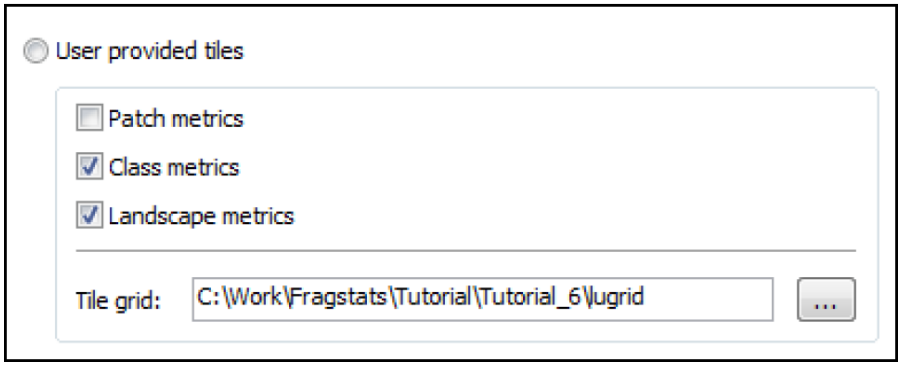
In our example, as shown below (left figure), the input landscape (lugrid) is subdivided into 40 tiles (sub-landscapes) representing townships. Each tile (town) has a unique integer-valued id ranging in value from 150 to 420 (note, the id's need not be consecutive). Each tile will be analyzed separately 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.

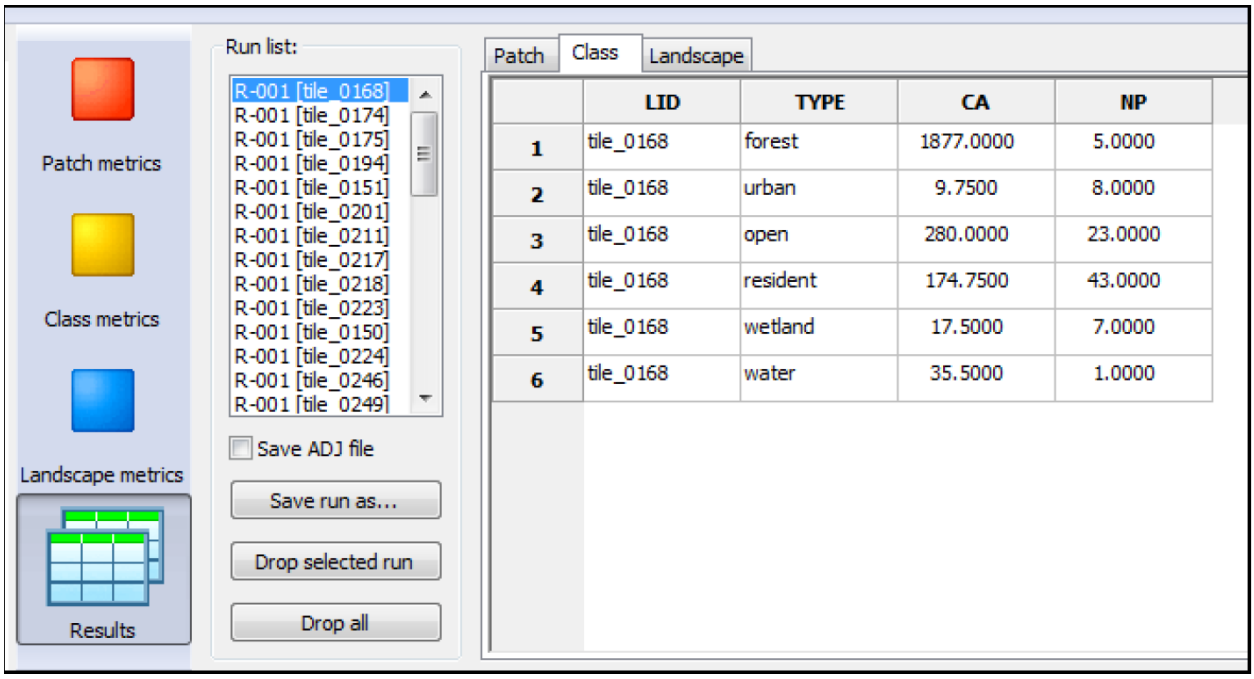


In addition, check the boxes for **Class** and **Landscape** metrics, as shown. Note, you must have at least one of these boxes checked or you will get an error message later when trying to run the model. However, only check the level corresponding to the metrics you want to compute.

Next, 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.



	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

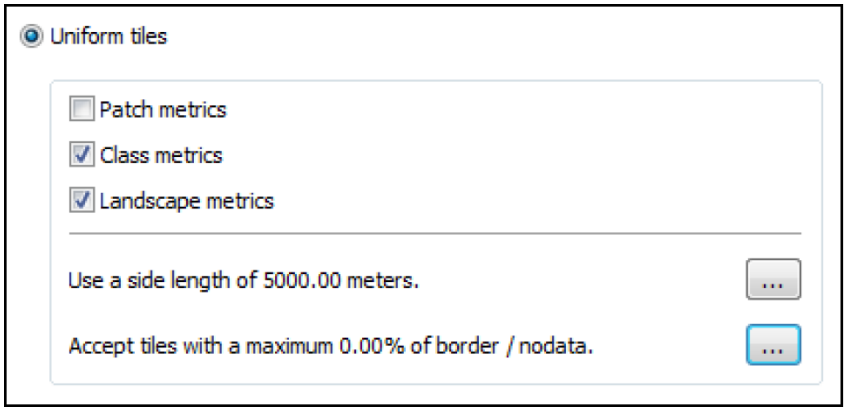
6.2 Uniform tiles

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.

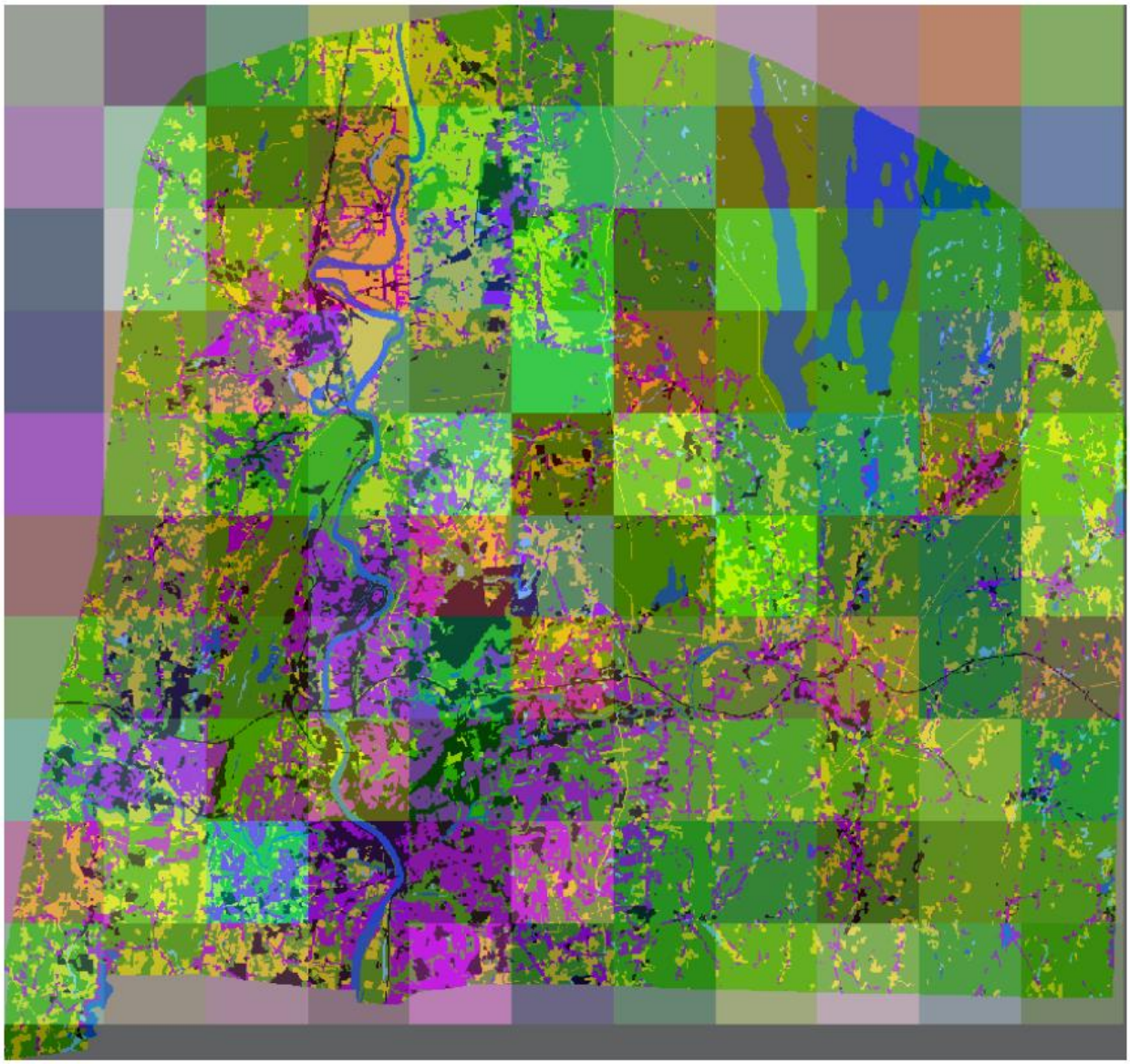
Next, you have the option of accepting tiles with a maximum user-specified percentage of border/nodata. The default is **0%**, which means that any tile that contains even 1 cell of either border (negative cells) or nodata will be discarded. Let's keep the default for now and see what happens.

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**).

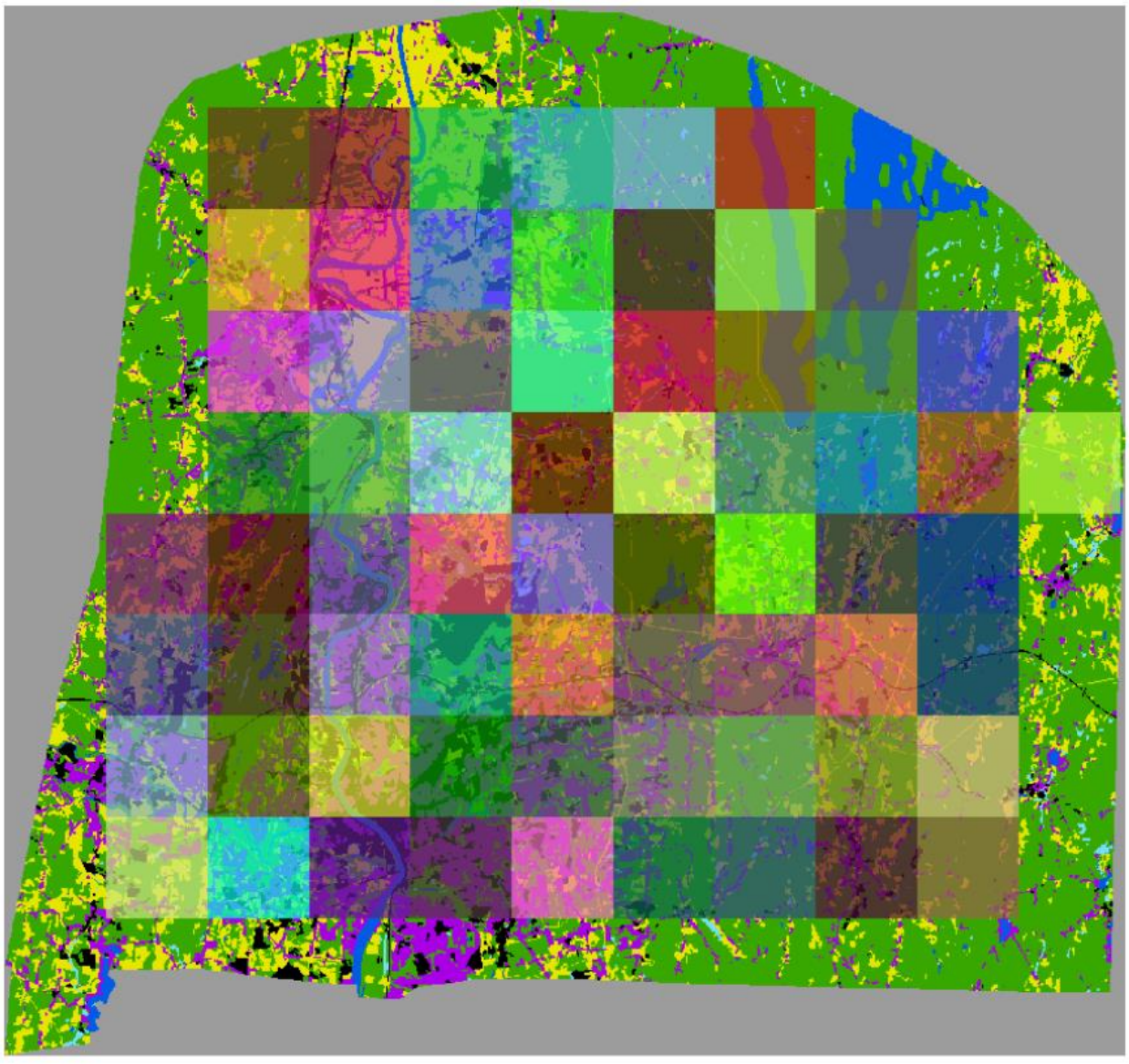
Open up the provided **fragtutorial_6.mxd** project in ArcMap, if it is not already open from earlier. Add the created uniform tile grid (**tiles00001.tif**) to the project. Note, each time FRAGSTATS creates a tile grid in the same directory, it will increment the tile grid name by 1. Since this is presumably 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 size 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.

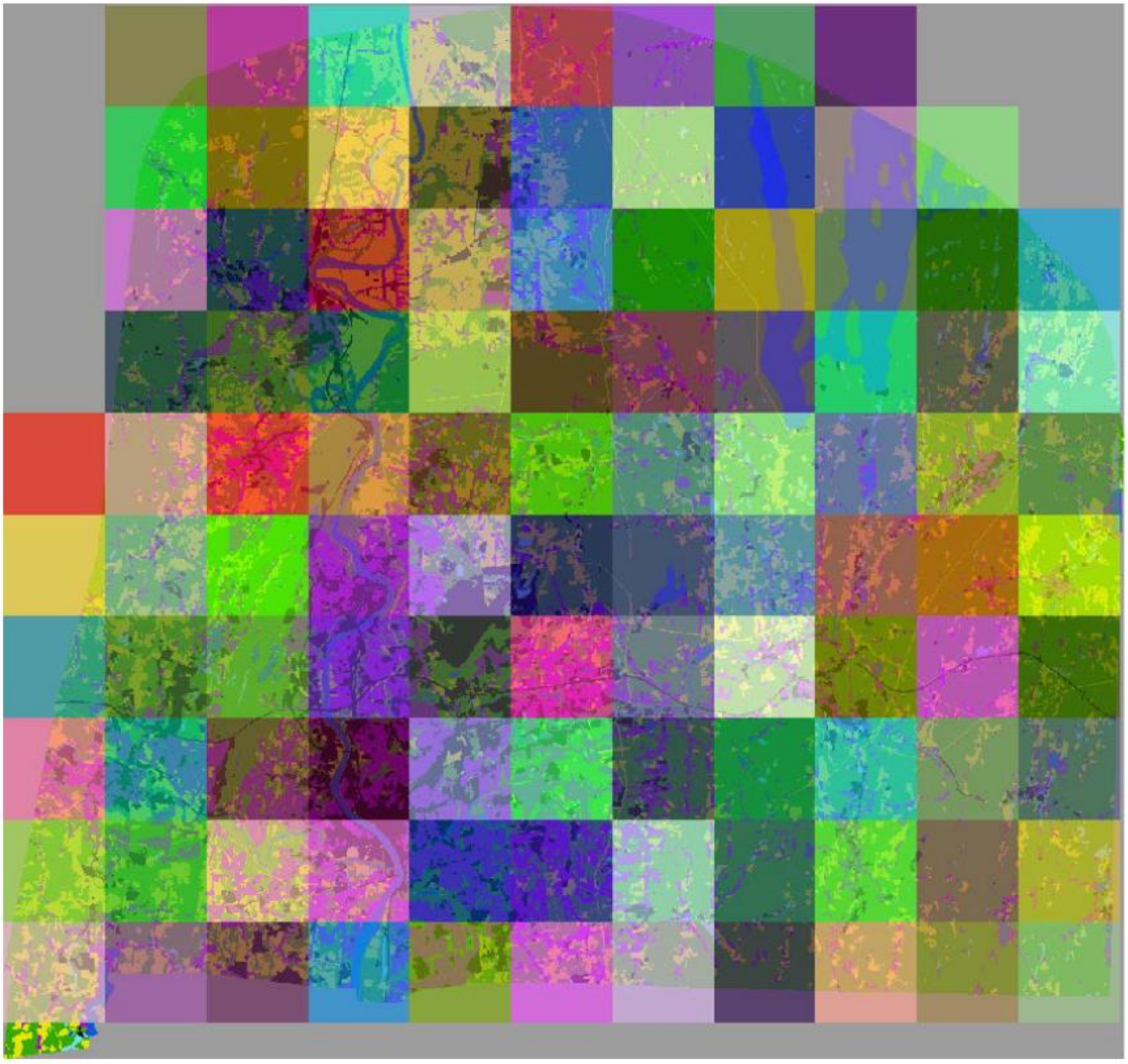
Importantly, because we specified a maximum threshold of 0% border/nodata, any tile containing even a single cell of nodata was discarded. In fact, the result summary



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.

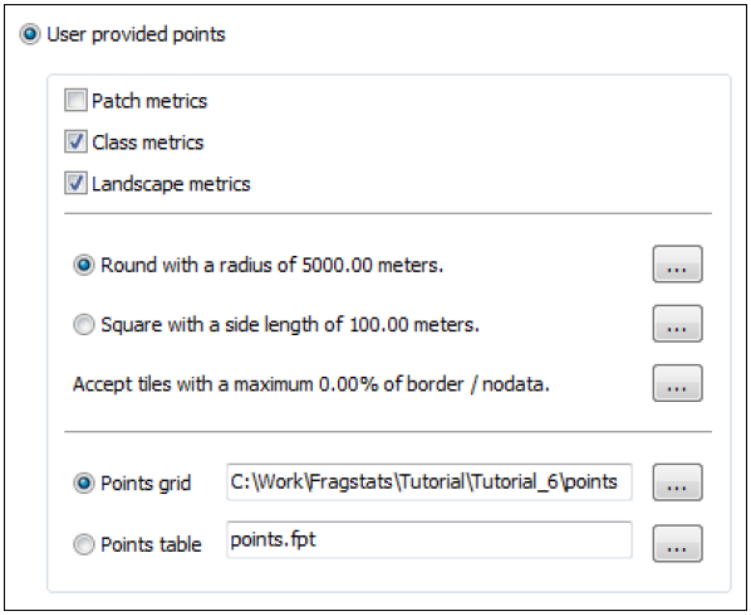
Now, let's change the maximum percentage of border/nodata from 0% to say 100% and see what happens. Note, a 100% doesn't actually mean that a tile that is composed entirely of nodata will be deemed valid, since this would be nonsensical. A valid tile still has to have at least one cell in the landscape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

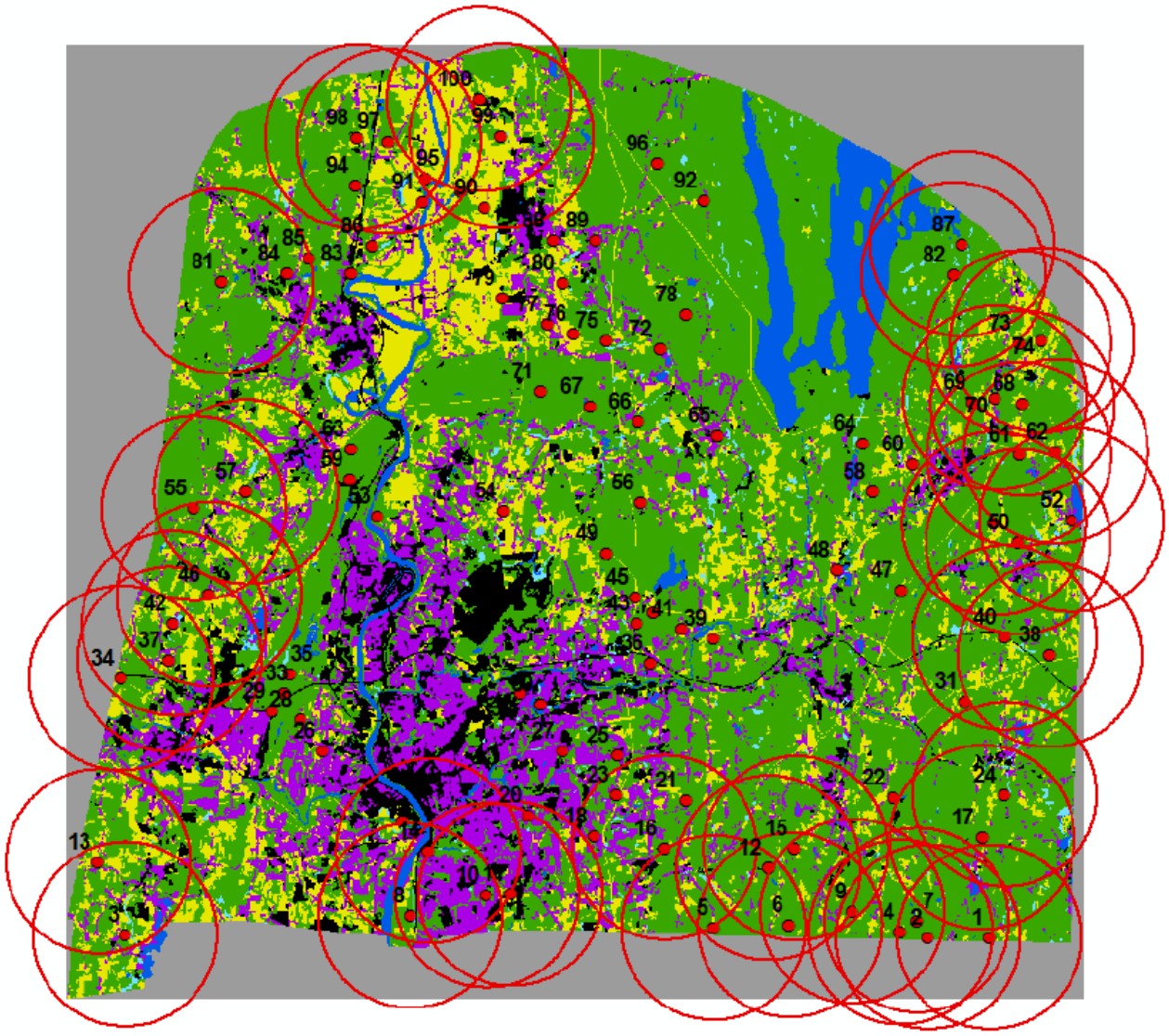
Choose either the points grid or points table to load by clicking on the corresponding radio button. To import the points grid (**points**), simply click on the **[...]** button and repeat the process for inputting a grid, making sure that the data type is the same as before. To import the points table, 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**).

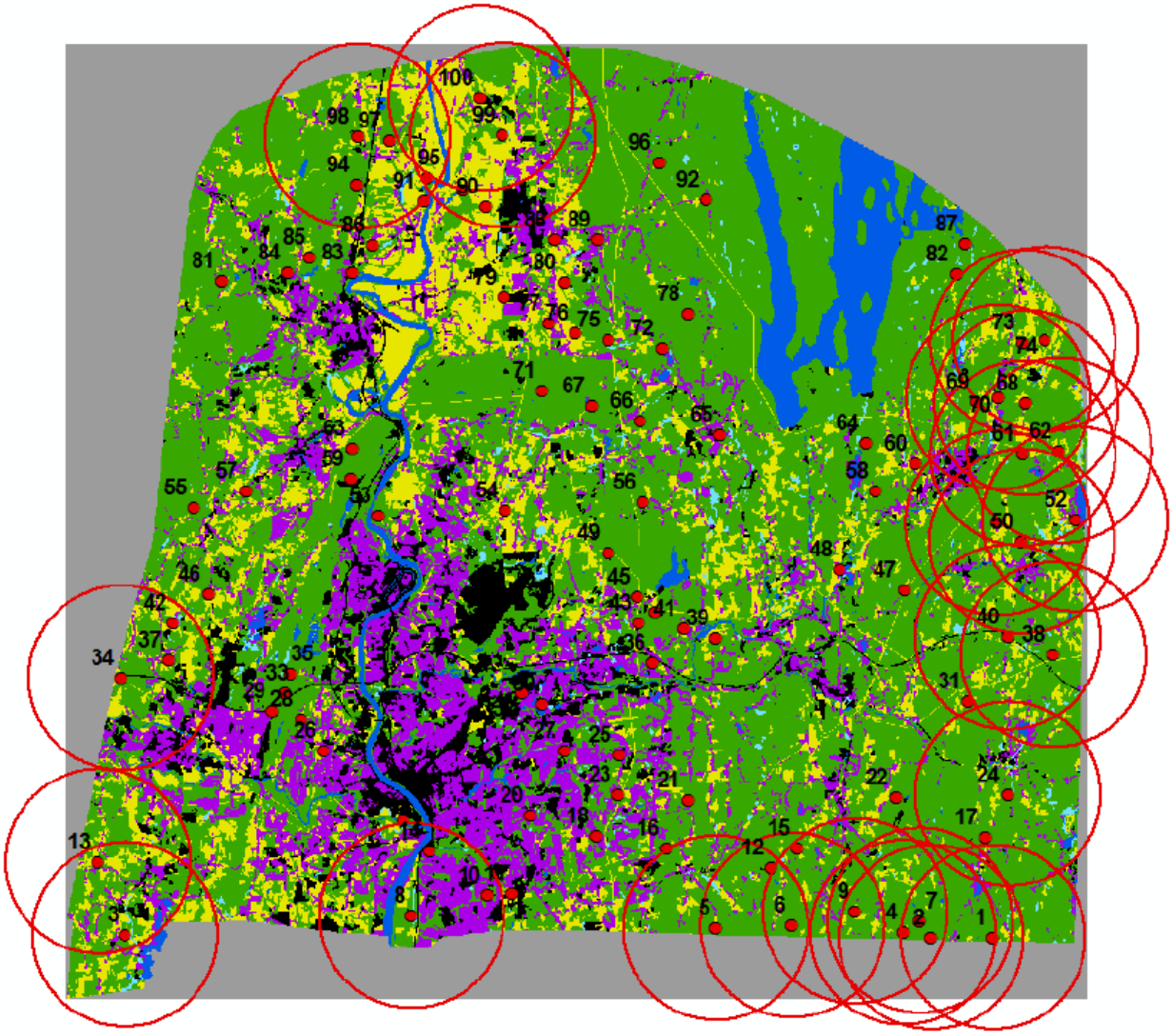
Open up the provided **fragtutorial_6.mxd** project in ArcMap, if it is not already open from earlier. Note, the ArcMap project contains the points grid (**points.tif**), as well as a points shapefile (**points shapefile**) included for the sole purpose of facilitating the display of the points. As shown 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.

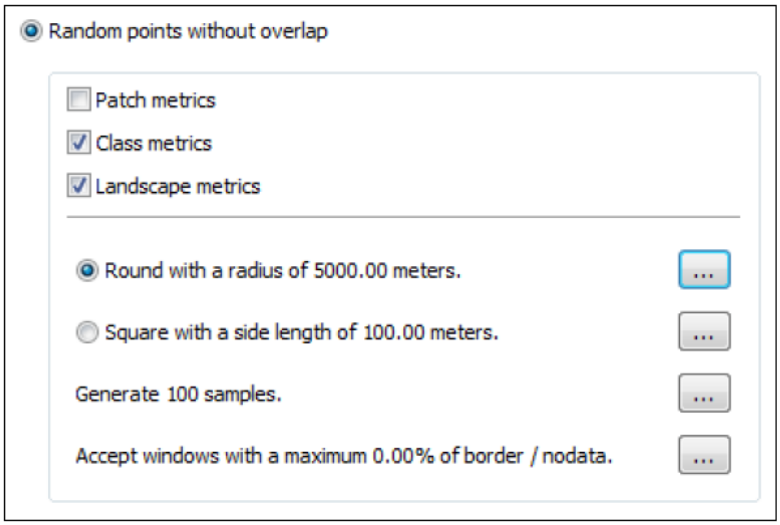
Change the threshold to **100%** and run the analysis and review the results. The run list (not shown here) should contain 74 rows, one for each valid window. The SUMMARY in the Activity log should indicate that there were a total of 74 windows, and that 74 were deemed valid. This is because the only 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.

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 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. 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

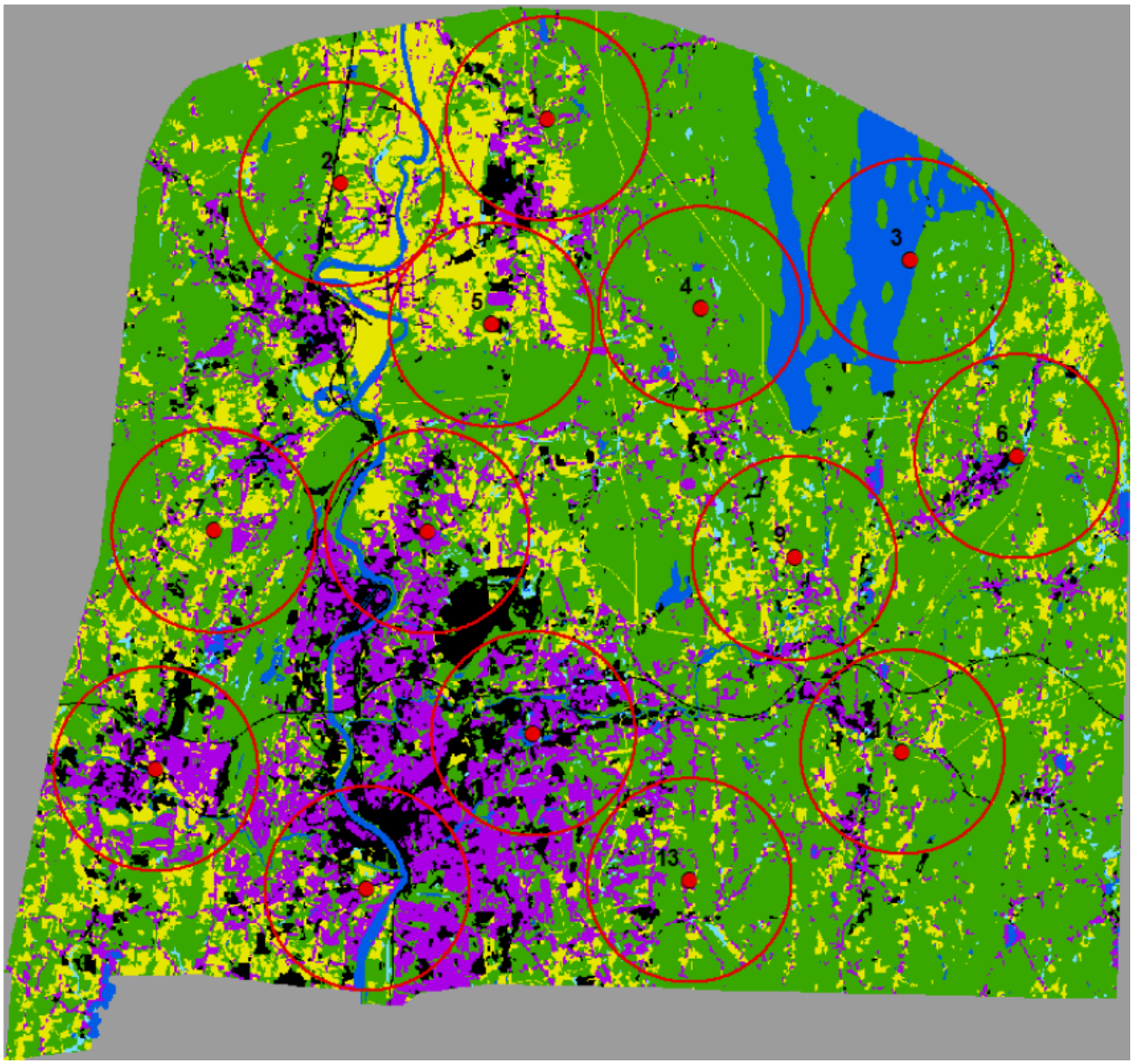
on OK at the bottom of the window and keep your fingers crossed that it doesn't crash.

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.

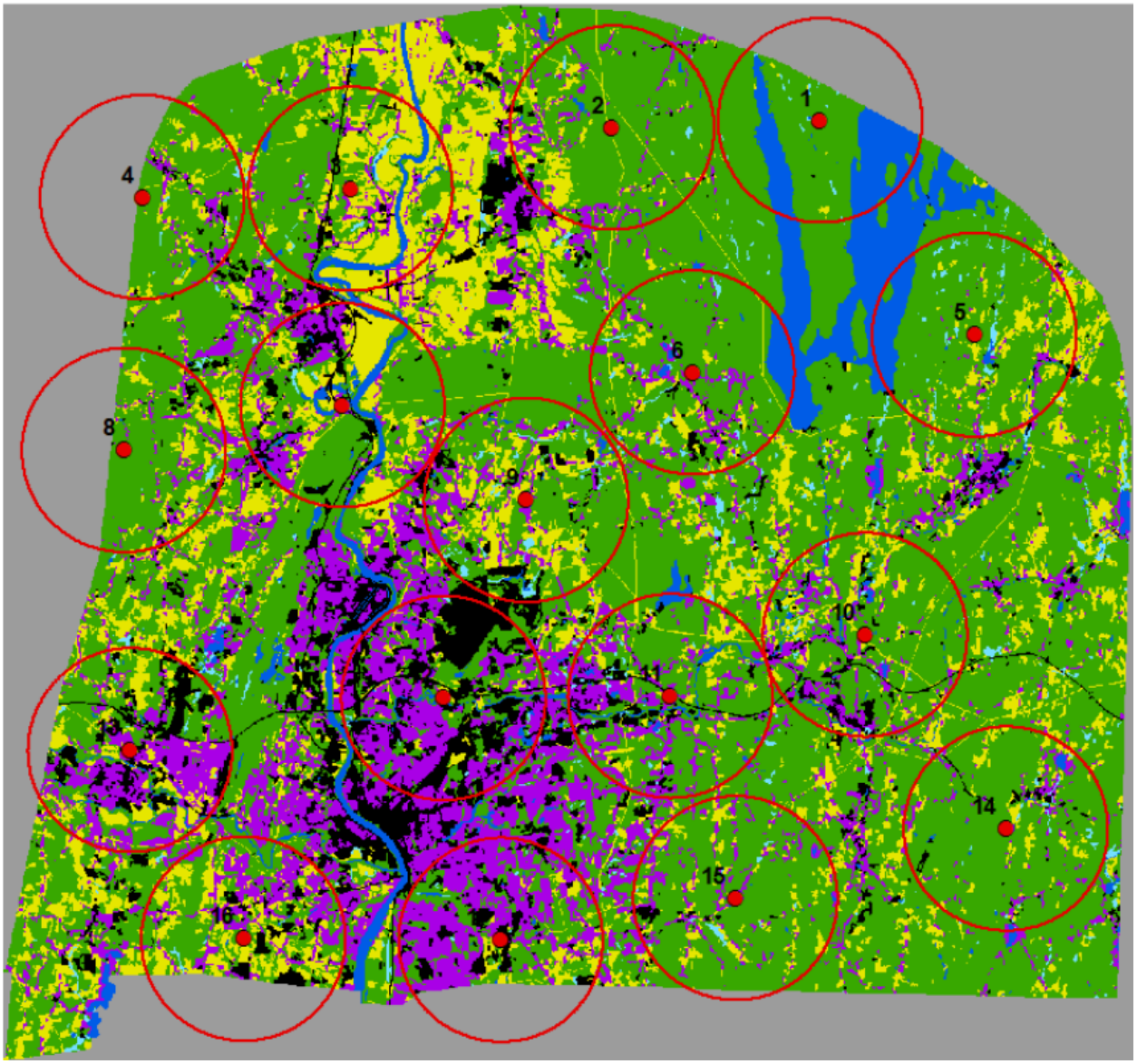
As shown below, in my particular run, 14 random points (windows) were generated out of the maximum desired 100 points. Note, the 5 km windows are all contained entirely within the landscape of interest (i.e., none of them include even a single cell of border/nodata because we specified a 0% threshold) 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.

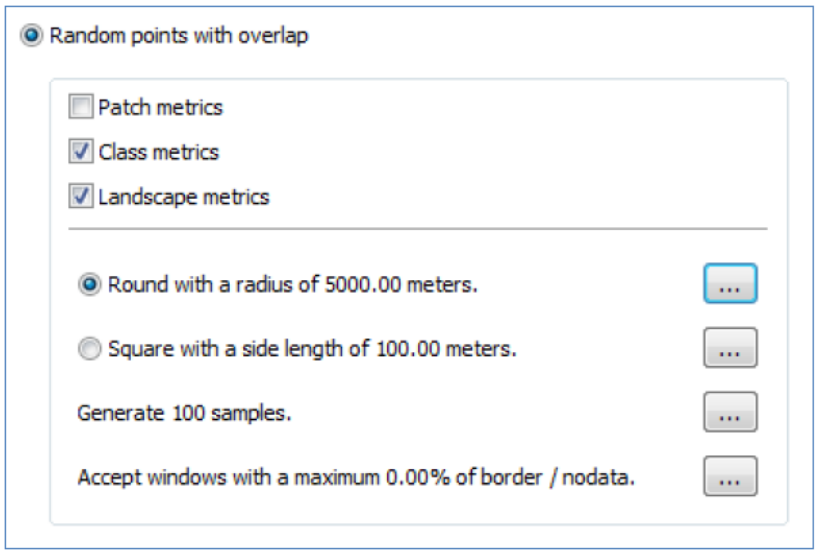


6.5 Random points with overlap

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.

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

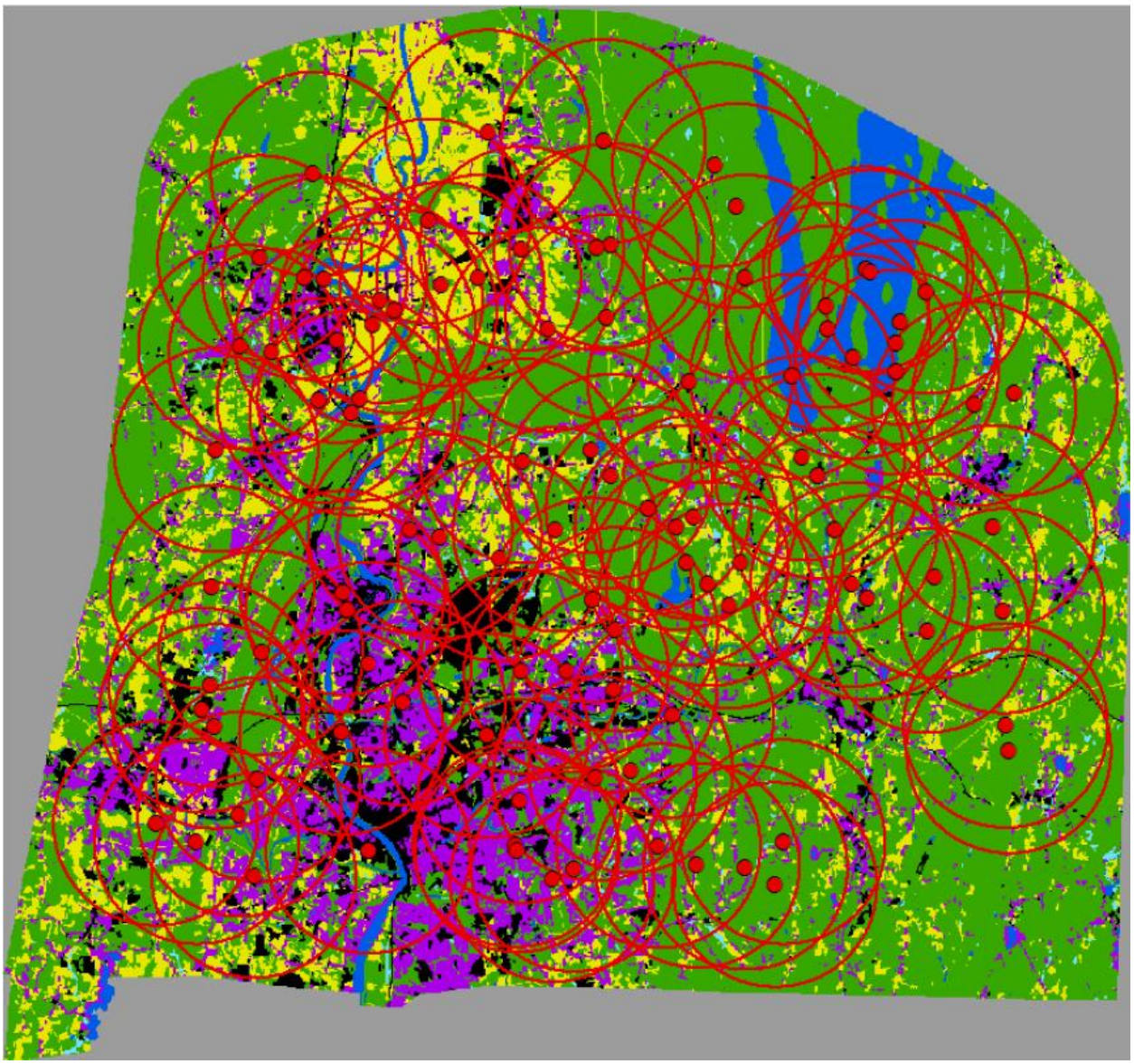
results pertaining to the random windows or sub-landscapes. In this case (but not shown here), the run list should contain multiple rows, one for each randomly generated window. The SUMMARY in the Activity log should indicate that there were a total of 100 windows generated. Note, in contrast to the 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.