Last Updated: 16 February 2023

Prepared by: Kevin McGarigal

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, not 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 and can 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')
```