Acoustic Primer Exercises

A Tutorial for Landscape Ecologists

Luis J. Villanueva-Rivera, Department of Forestry and Natural Resources, Purdue University

Bryan C. Pijanowski, Department of Forestry and Natural Resources, Purdue University

Jarrod Doucette, Department of Forestry and Natural Resources, Purdue University

Burak K. Pekin, Department of Forestry and Natural Resources, Purdue University

This tutorial is a supplement to the paper *A Primer on Acoustic Analysis for Landscape Ecologists* by Villanueva-Rivera et al featured in the Landscape Ecology special issue entitled "Soundscape Ecology" (vol. 26, pages 1233-1246). Accordingly, the exercises in the tutorial are meant to be undertaken while reading the article.

Data and links to tools used in this tutorial can be found at http://ltm.agriculture.purdue.edu/soundscapes/primer

ACKNOWLEDGEMENTS

This tutorial was first published on February 6, 2011 as version 0.9. This version (1.2) was published on January 9, 2013.

CONTENTS

EXERCISE 1 – LISTENING TO AND VISUALIZING THE SOUNDSCAPE

Inspecting data in its raw format before deciding on the type of analyses to conduct is an important part of effectively assessing and understanding the system which one is studying. Accordingly, this first exercise is designed to develop in the reader an intuitive sense of the sound data that is quantitatively analyzed in the following exercises. It also demonstrates the sounds produced by different objects (e.g. biophony, anthrophony) and their interactions within the soundscape. The reader should consult 'exercise 1'section of the manuscript by Villanueva-Rivera et al. (this special issue) for a description of the sounds they are hearing as well as the information conveyed by them.

PART A. LISTENING

GETTING STARTED

1. Visit the website ltm.agriculture.purdue.edu/soundscapes/primer to listen or download the three sound files in this exercise.

LISTENING TO SOUNDSCAPE RECORDINGS

2. Click on exercise 1. Play sound example #1 by clicking play button on the left bottom corner. You will need a modern browser with the Adobe Flash plug-in installed. You will see this web page with
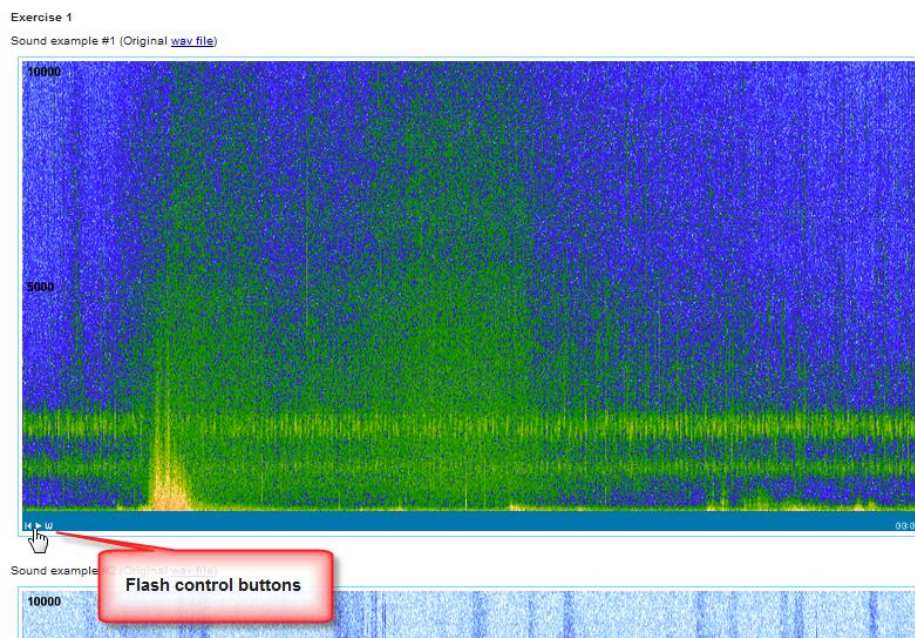
You can use the control buttons on the lower left of the window to toggle between spectrogram (labeled as S) and waveform (labeled as W). When you toggle to waveform and you would see this:
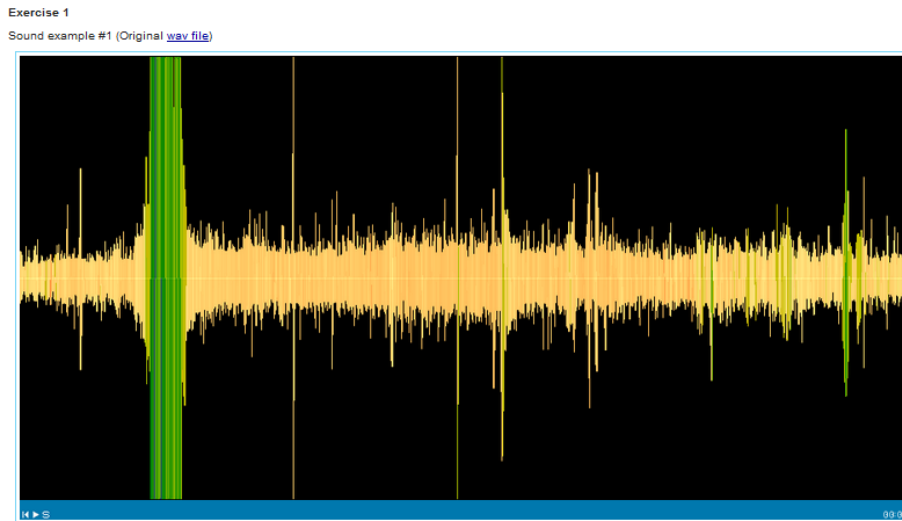
You can also select the arrow to play the recording. Once the file loads over the internet, you can also move along the sound file using the slider:

3. Repeat for sound example #2 and sound example #3. Watch the spectrogram as you listen, and note the region on the spectrogram that corresponds to the sounds you are hearing.

PART B VISUALIZING SOUND

GETTING STARTED

In this exercise, we will use Raven Lite (Charif et al. 2006) and Audacity to examine several soundscape recordings from the Tippecanoe Soundscape Study. Raven comes in several versions, the freeware version is called Raven Lite (currently at version 1.0). Users can acquire a free license from Cornell after they download and install Raven Lite.

Raven Lite can be downloaded from http://birds.cornell.edu/brp/raven/Raven.html, where versions for the Windows, Mac OS X, and Linux operating systems are available. After the software has been installed, you can go to the Help menu and then Get Free License. You will be directed to a web site and then requested to "purchase" the software. An email will be sent

with your license code. Directions for providing the license number to the program will be provided in that file.

Audacity can be obtained from http://audacity.sourceforge.net, where versions for the Windows, Mac OS X, and Linux operating systems are available, as well as the source code.

At our tutorial web site are several wav files that will be used. Download exercise2.zip and place the files (unzipped) into a folder called Sounds101_demo.

OPENING AND VIEWING A SOUND FILE IN RAVEN LITE

Let us open a small 5 sec sample of the May 14, 2008 Purdue Wildlife (wetland) recording from 01:00 (1:00am local time). This sample contains chorusing of Spring Peepers during a light rain shower. The interface to open a file will look like this:
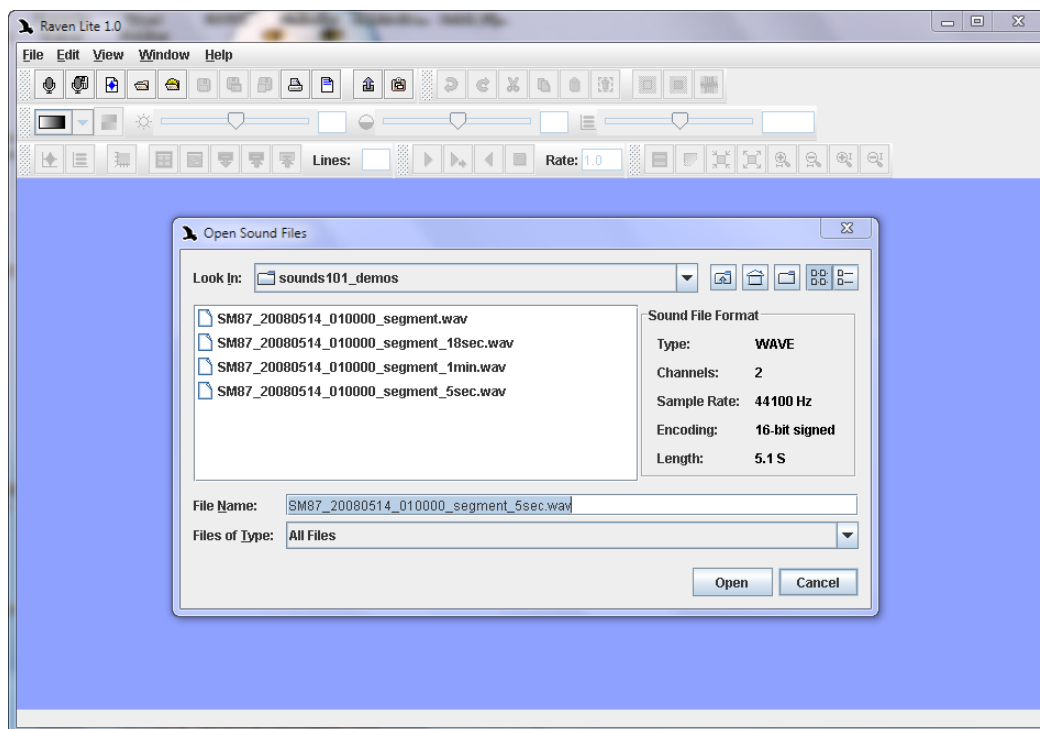
Once this file is selected, click the Open button and select Channel 1. The sound file will be loaded and a waveform and spectrogram shown like this:
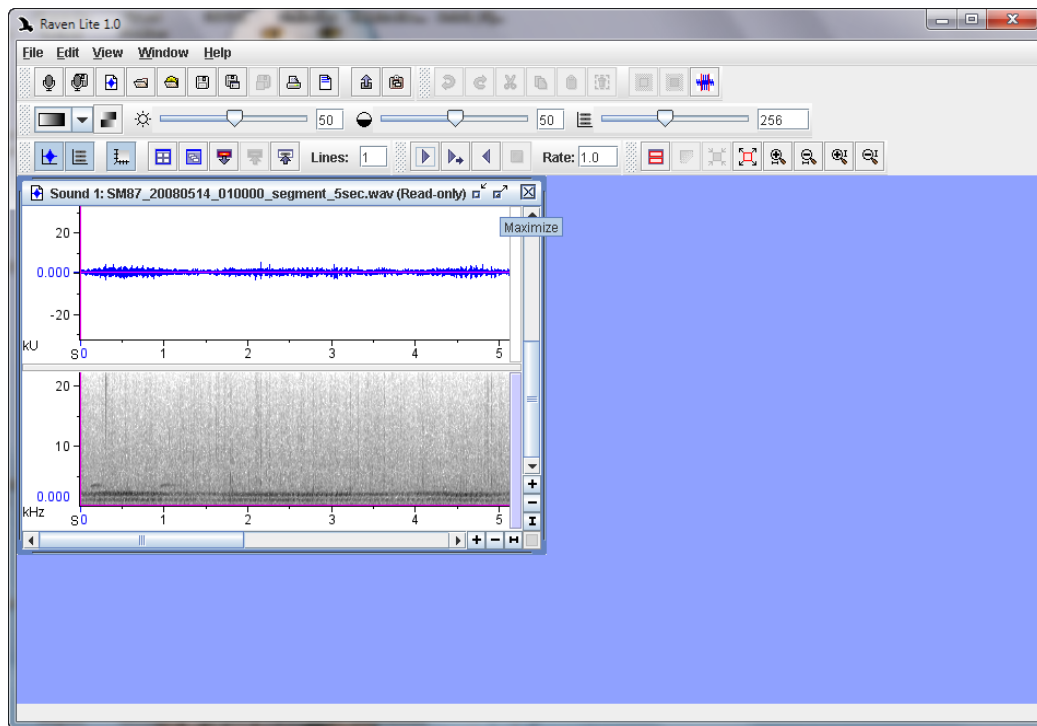
You can select the Maximize button
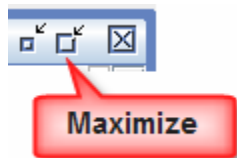
To create a large view of both the waveform and spectrogram:
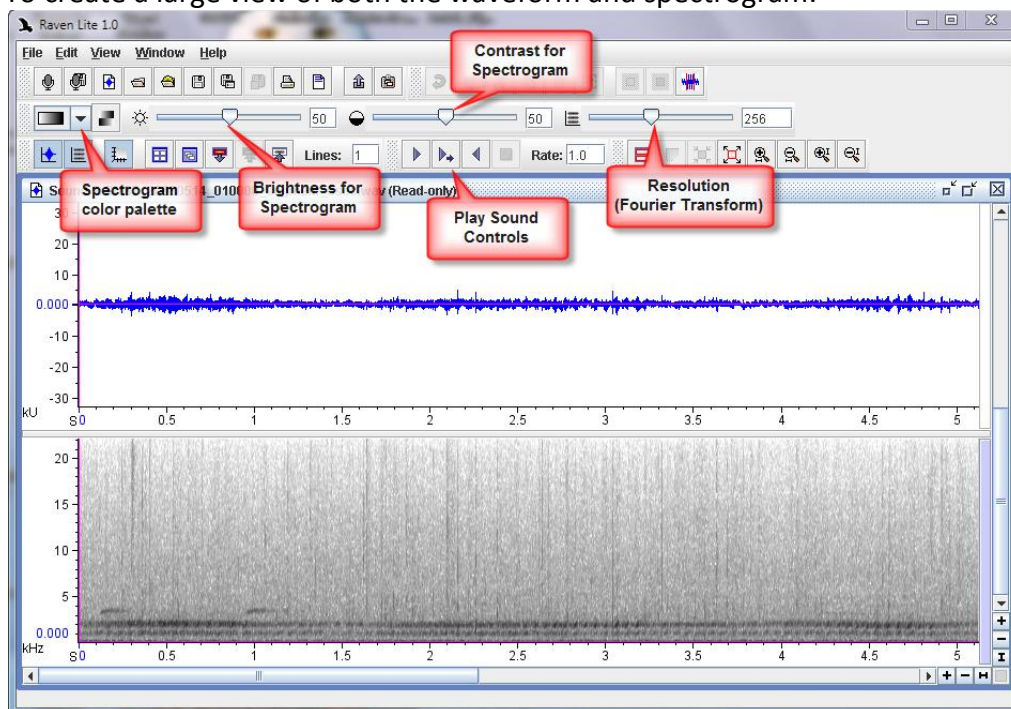
Note that there are several sets of buttons that control various functions related to visualizing, subsetting, file management, and zooming. The spectrogram color palette provides four color schemes for the spectrogram. The brightness, contrast and resolution have scroll bars controlling these three key features of the spectrogram. Change each to see how they modify your ability to see patterns within the spectrogram. Also note that you can also play the sound using the three play sound control buttons (play, play and scroll, and play backwards). After experimenting with the three main spectrogram visualization scroll bars, play the sound several times. You should hear the Spring Peepers chorusing during a light rainshower.

SUBSETTING SOUND IN RAVEN LITE

Raven Lite allows a user to select areas within the sound file and then remove other sounds so that you can hear specific portions of the spectrogram. Use your cursor to select sounds within the first 2 seconds that are less than 3 kHz. The selection should appear as a red box with portions highlighted with small squares:
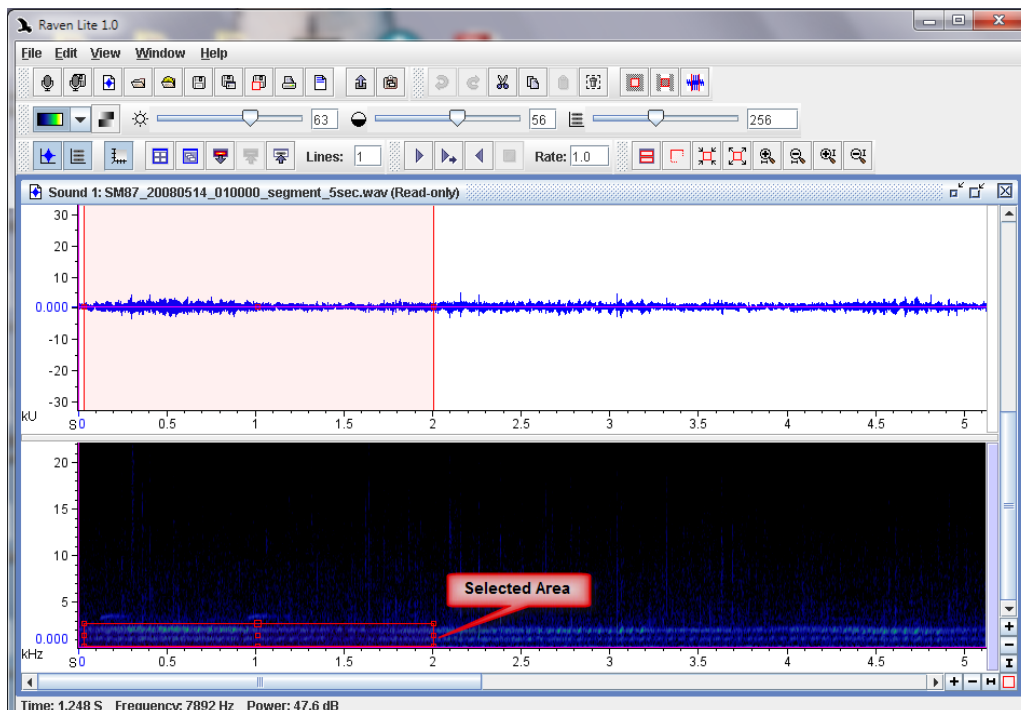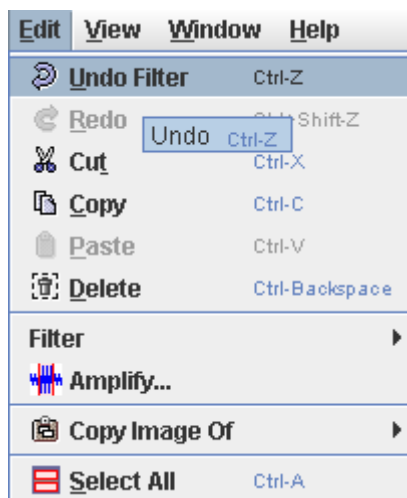

Using the Filter Around tool,
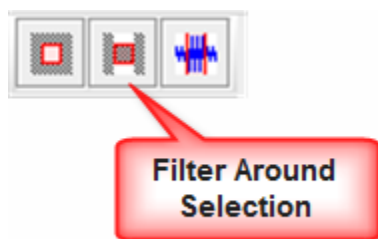
you will see the sounds outside the selected area set to 0 dB:
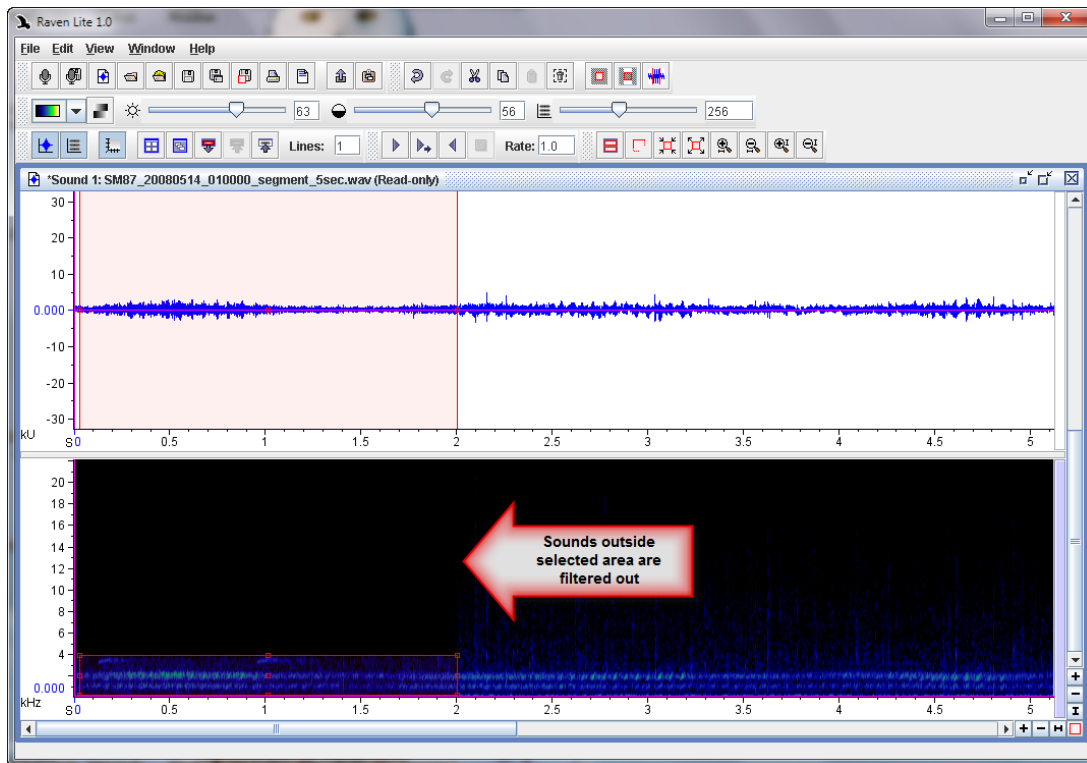
Play the sound now and compare this to the situation where the file does not have a filter applied. You can toggle between having the filter applied and not applied using the Edit menu (Undo Filter and Redo):


You may also Amplify the sounds that are selected. Select Amplify from the Edit menu and then key in a value of 4 (values less than 1 reduce the sound volume) and then listen to the recording again.

Readers interested in learning more about Raven Lite or the other Raven products can go to the Cornell Laboratory for Ornithology web site listed above. A Raven Lite tutorial is also available as well as access to thousands of files to listen to many individual vocalizations.

VISUALIZING SOUND FILES USING AUDACITY

Start Audacity and bring up the same 5 min recording used in the Raven Lite portion of this exercise. Doing so will provide you with this:
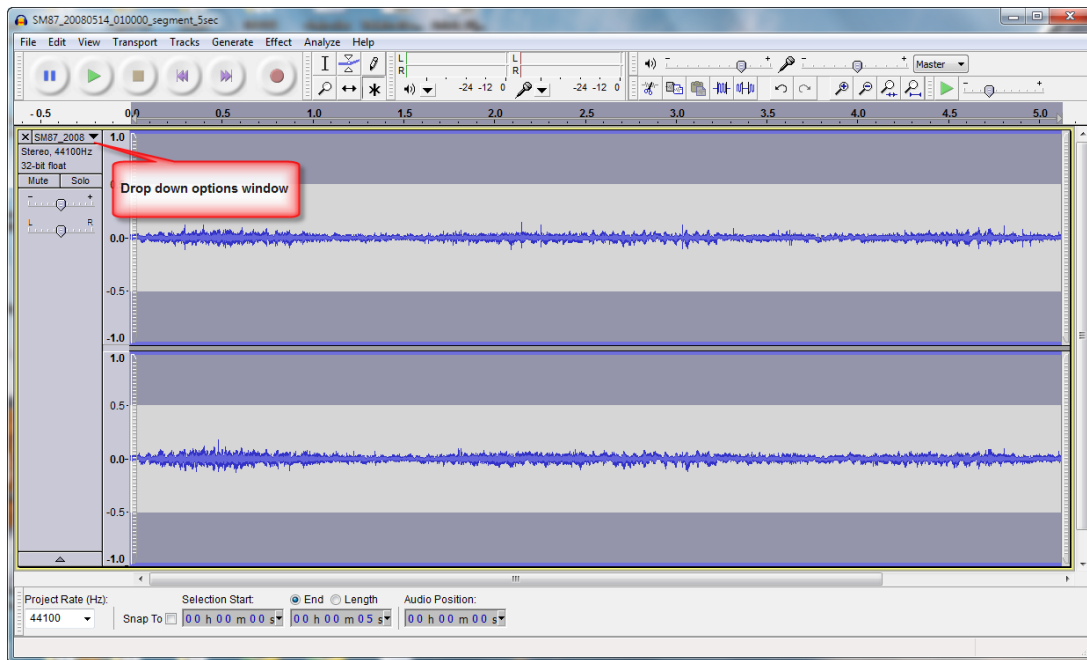
Because this is a stereo recording, you will see two waveforms, one for each channel. You can use the drop down options menu to toggle what you see between spectrograms and waveforms. A spectrogram view would show this:
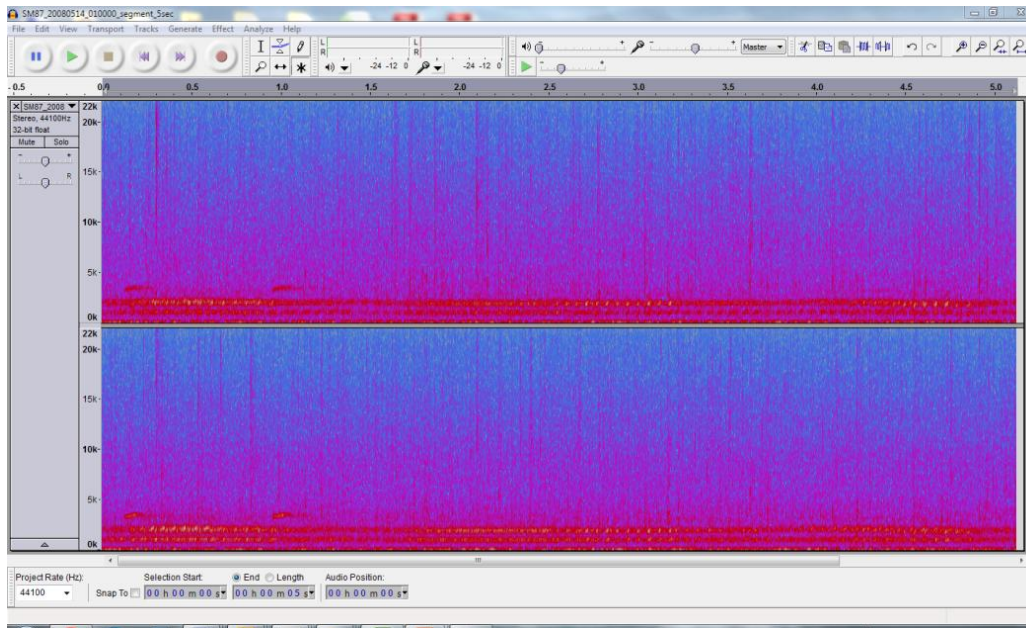
You can control the playing of the sound using the large menu of buttons in the upper left.

Play Sound
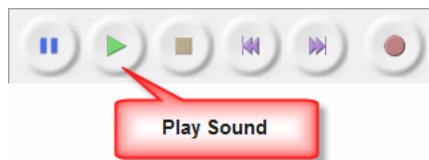
You can create a frequency-amplitude plot of this 5-min recording using the Analyze and Plot Spectrum options:
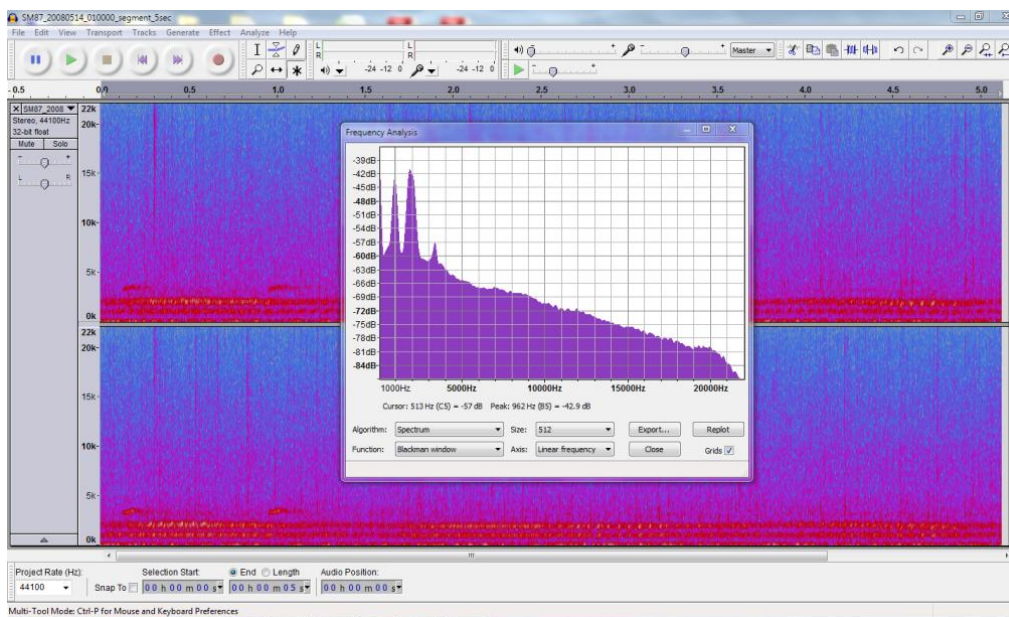
EXERCISE 2 – FREQUENCY BAND ANALYSIS

In this exercise, we will use the R packages *seewave* (Sueur *et al.* 2008) and *tuneR* (Ligges 2009) to analyze sound files. R is a free software environment for statistical computing and graphics which can be downloaded from http://cran.r-project.org/.

GETTING STARTED

R comes in two versions, compiled binaries and source code. We will use the compiled binaries. The main web page shows how to acquire either version. Select the operating system that you will be using (e.g., Windows) and then select the base binary option. We used R version 2.12.1 for this tutorial and versions 1.5.9 of *seewave* and 0.3 of *tuneR*.
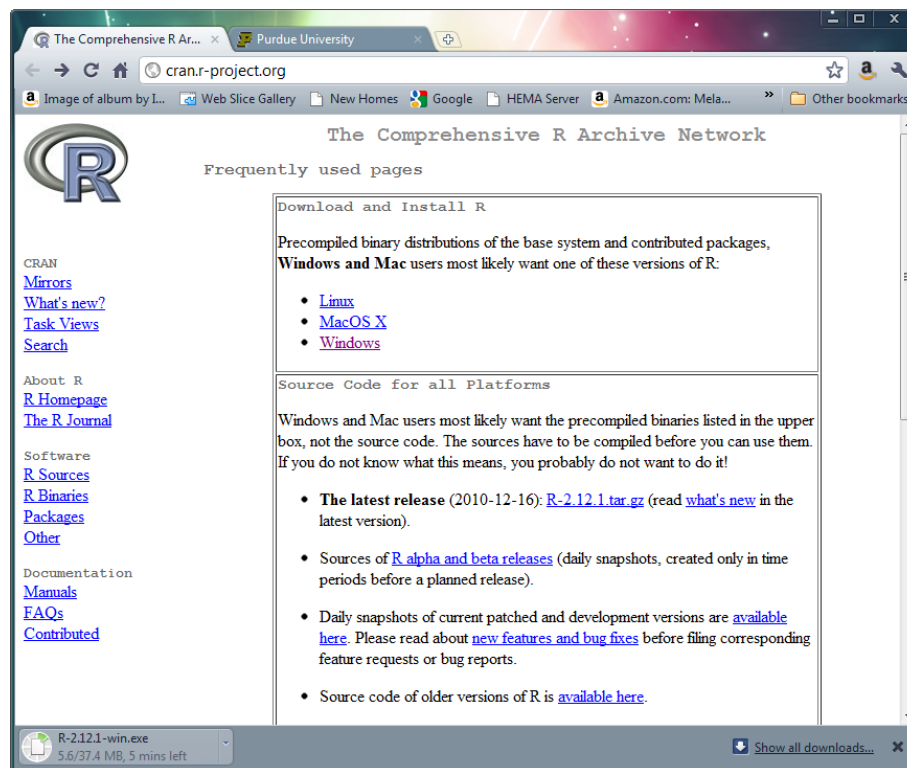
INSTALLING AND LOADING PACKAGES IN R

While R comes with many of the packages needed for basic statistics already installed, the package *seewave* will need to be installed manually.

1. Once you have downloaded and installed R on your computer, open R. You should have a window that looks like this:
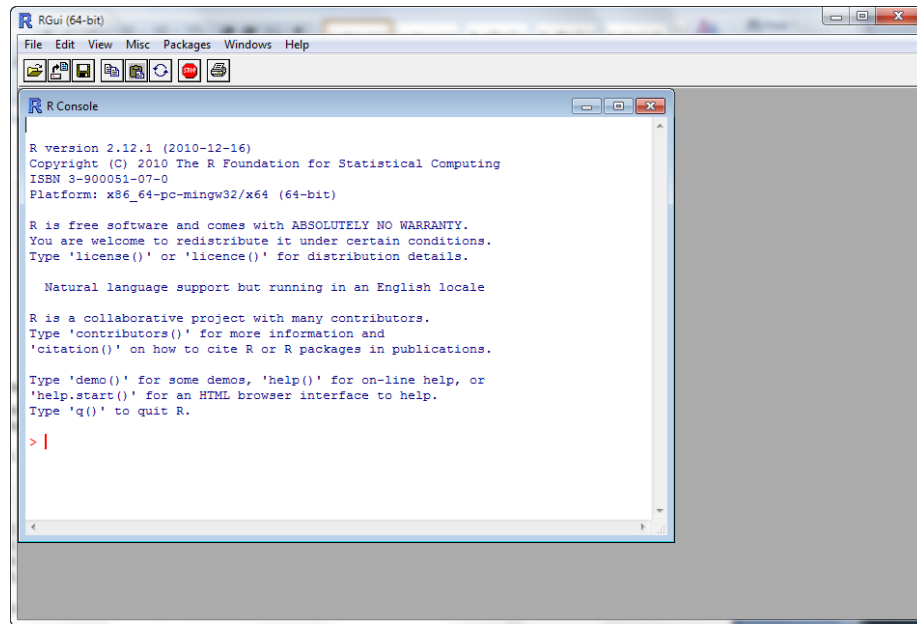
2. Click the **Packages** tab at the top, and then **Install Packages**.
3. A separate window called "CRAN mirror" will pop up. Select a nearby location and click ok.
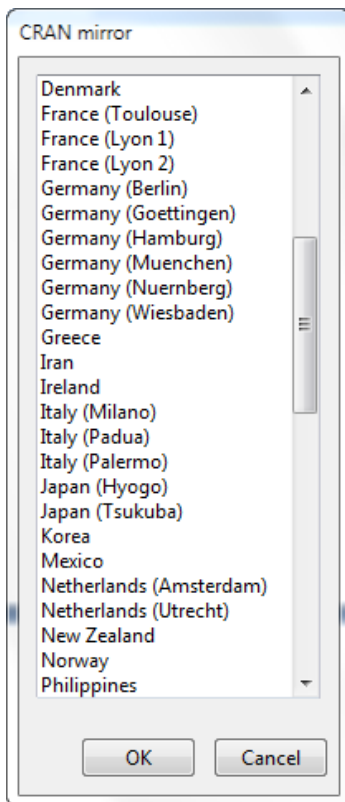
4. The packages window will open. Scroll down until you find *seewave* and click ok.
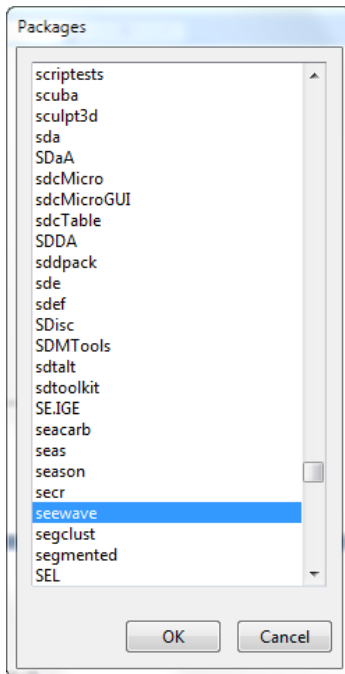
5. Now load *seewave* by clicking the **Packages** tab again, and selecting *seewave* from the list and clicking ok. The package *tuneR* will be installed automatically with *seewave*.

DOWNLOADING DATA FILE AND SETTING WORKING DIRECTORY

You will need to download the sound file data which is in .wav file format and load it into R.

1. Go to ltm.agriculture.purdue.edu/soundscapes/primer and click on exercise 2.
2. Right click on wav file for sound example #1 and save to folder on your computer.
3. Right click on wav file for sound example #2 and save to the same folder on your computer.
4. On R, click the **File** tab, and then **Change dir**.
5. Select the folder you saved your sound files in from the window that pops up. Click ok.

RUNNING R ANALYSIS PACKAGES

To load the file as an object[1], in this case called *sound1*, the *tuneR* function `readWave()` can be used

1. Copy and paste the following line into R console (note that R is case sensitive);

```
sound1 <- readWave('file1.wav')
```

---

[1] An object is a computer programming term for an identity of a file or other "object" that can be manipulated by the program. Here, sound1 is an object associated with the wav file that is now loaded into the program R.
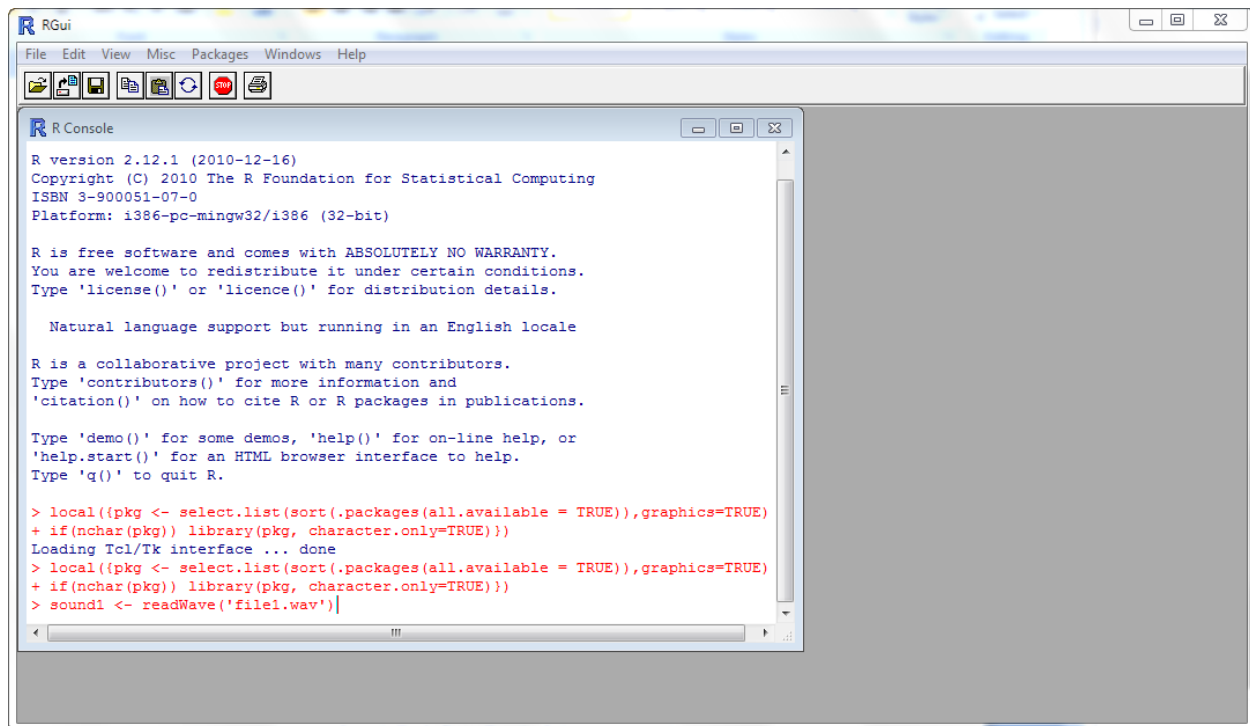
2. To see the spectrogram of the sound file, use the *seewave* function *spectro()* to display a FFT window of 512. Make sure you maximize the R Window. Copy and paste following line;

```
spectro(sound1, wl=512)
```

The following graphic is produced from the above command.
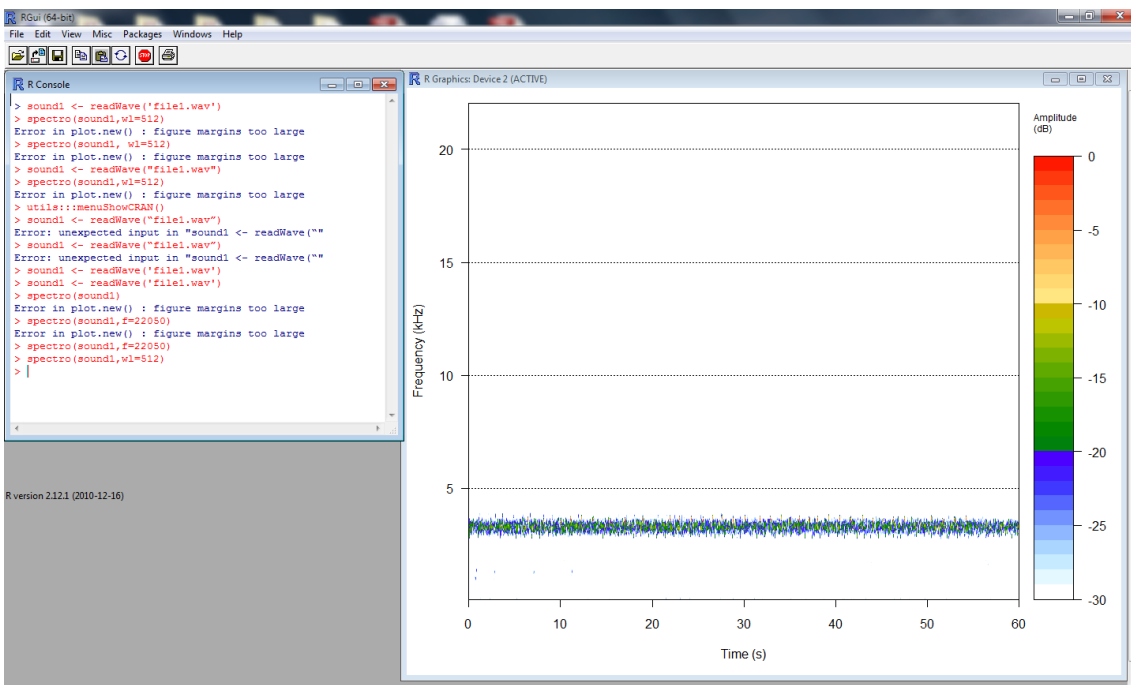
By default, the spectrogram does not show the fainter sounds. The option *collevels* lets you specify the levels at which to draw the sound intensity of the spectrogram. In this case we will specify a sequence of levels between -70 and 0, with steps between levels of 5, using the function *seq()*. In addition, by setting the option *osc* as TRUE indicates seewave to draw a waveform under the spectrogram:

```
spectro(sound1,wl=512,collevels=seq(-70,0,5),osc=TRUE)
```
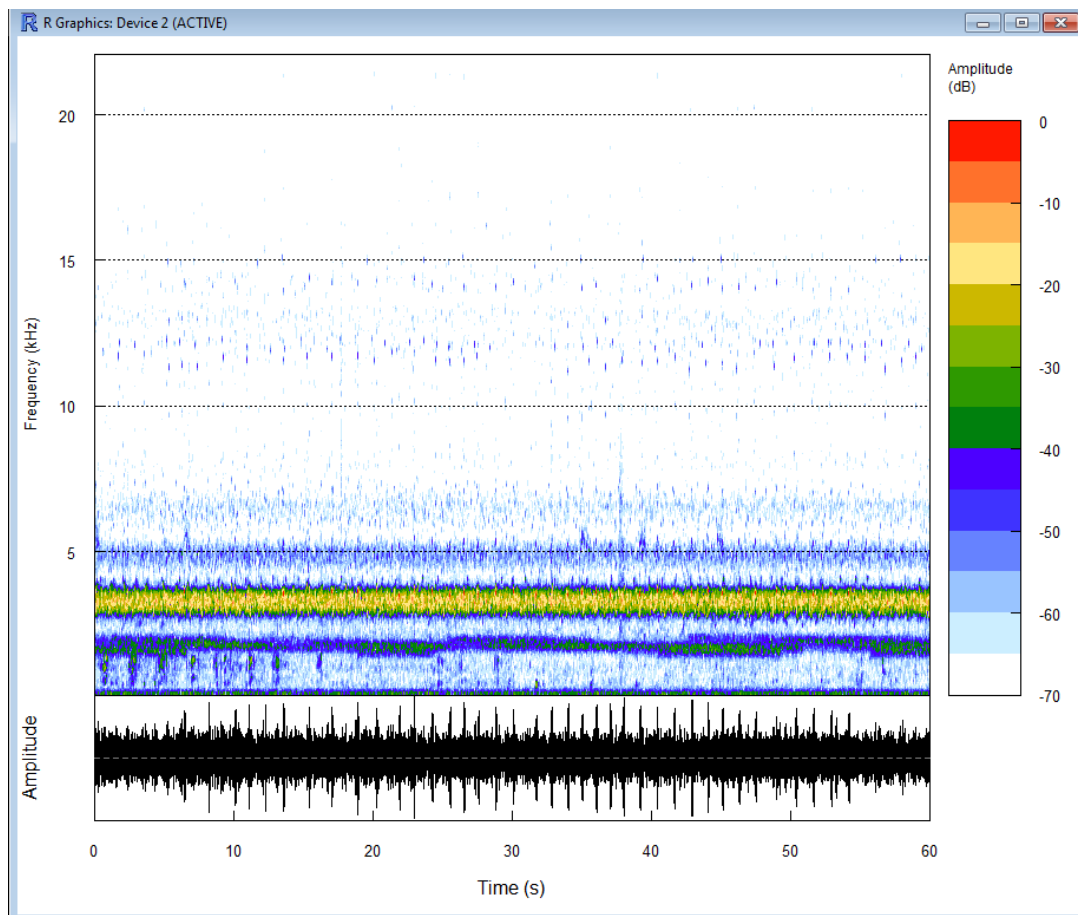

The function *spectro* can also save the data of the spectrogram as a numeric matrix with amplitude values, with each column being a Fourier transform of the window length *wl*. In this case, we assign the result of the function to a new object, in this case *data*, and setting the value of *plot* to false:

```
data <- spectro(sound1, wl=512, plot = FALSE)
```

2. Spectrogram matrix of values and divides it in 10 bands of 1 kHz each, up to 10 kHz. A default cutoff value of -50 dBFS is specified to remove the faint sounds. The proportion of sounds that falls above that cutoff value is determined in each band. We use this matrix to calculate an acoustic diversity index (Pijanowski et al. 2011, Villanueva-Rivera et al. 2011) using this script which is posted on this tutorial web site. Shannon's diversity index and the Gini coefficient is calculated for each channel (left and right microphones).

PROCESSING OF TIPPECANOE SOUNDSCAPE STUDY RECORDINGS

To demonstrate how this script works, we use two sound files that were recorded in the Purdue Wildlife Area (Figure 10 in Villanueva-Rivera et al. 2011) on 20 April 2008. The first sound file was recorded at night, at 00:00, and the other one during the morning, at 07:00. The night sound is dominated by frogs in the 1.3-3.8 kHz range while the sounds in the morning are from birds that range from approximately 0.2 kHz to almost 8 kHz (Figure 10 in Villanueva-Rivera et al). Just by visually comparing the spectrograms of these sound files, there is a larger diversity of sounds in the morning sound file that in the night sound file. We show how users can execute two scripts, one that processes a single file and another designed to batch many .wav files so that values can be plotted over time and at multiple sites.

1. Single file processing. From the R graphical users interface (GUI), select 'Open script...' and open the script file *soundscape_band_diversity.R*, then from the 'Edit' menu, select 'Run all' to run the script. If working from the command line, load the script by typing:
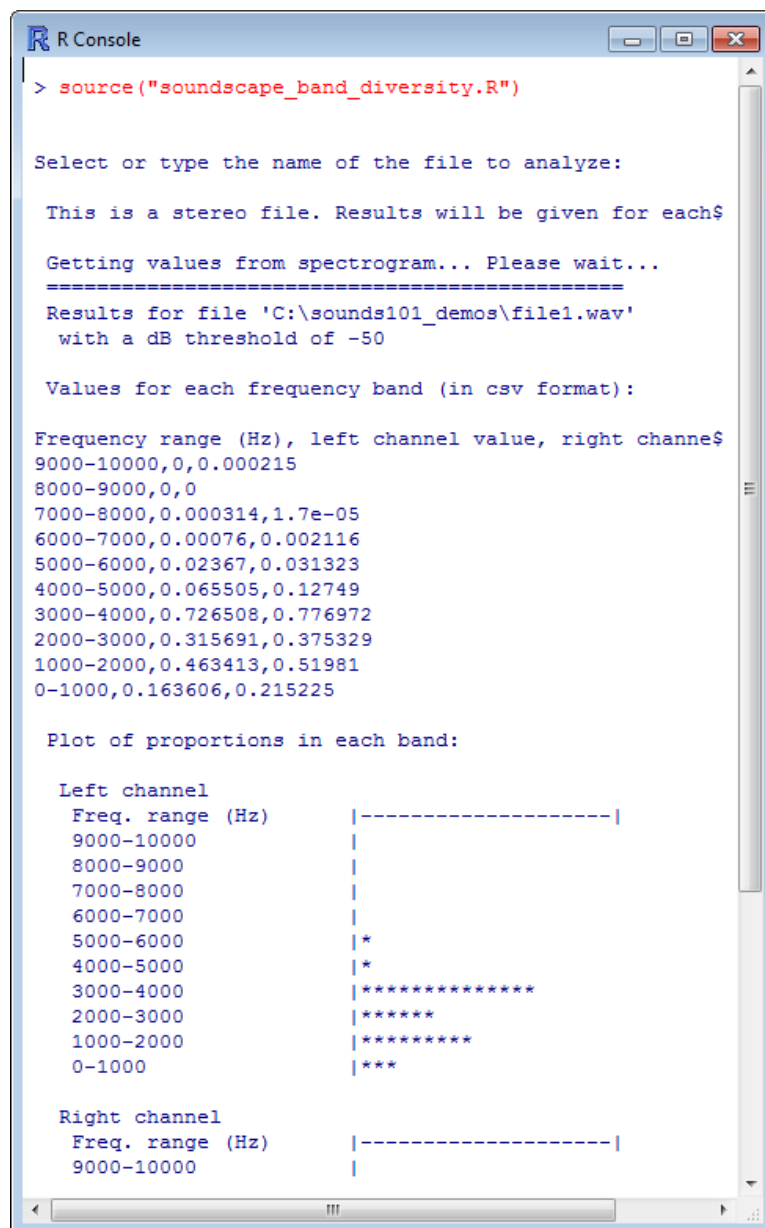
```
source("soundscape_band_diversity.R")
```

The script will ask for a .wav file. The dialogue box that will appear will look like this:
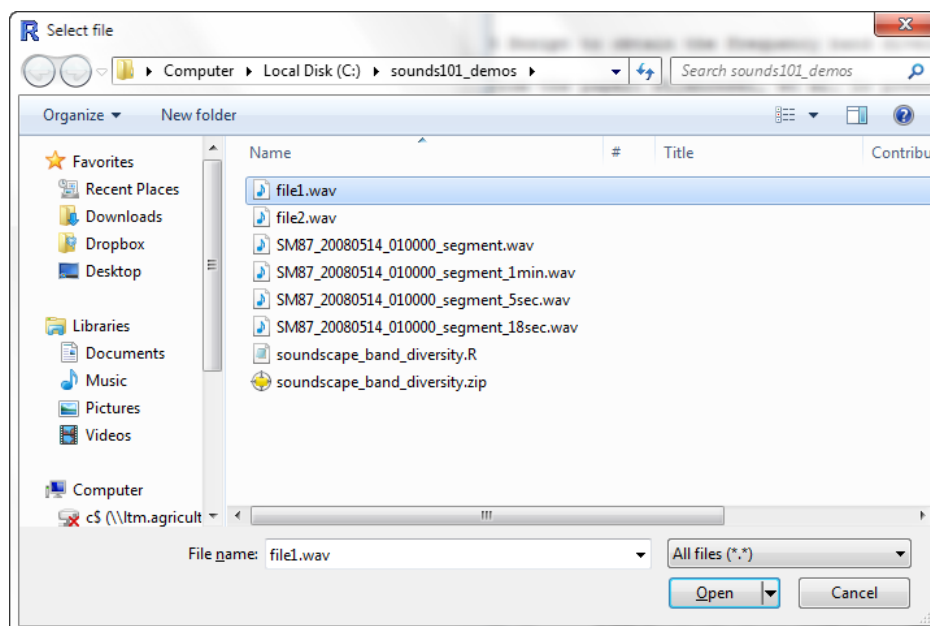
In this example, the night recording has most of the signal in the bands corresponding to 1-4 kHz, while the signal for the morning has strong components from the 0 Hz up to the 6 kHz bands. In addition, the morning sound file had higher proportion values in all bands. These patterns result in a higher frequency band diversity index for the morning sound file, with a value of 0.203, compared to the night sound file, with a value of 0.148.

Output from the script will be written to the screen like this:

```
R R Console                                          [-][□][✕]

> source("soundscape_band_diversity.R")


 Select or type the name of the file to analyze:

  This is a stereo file. Results will be given for each$

  Getting values from spectrogram... Please wait...
 ================================================
  Results for file 'C:\sounds101_demos\file1.wav'
   with a dB threshold of -50

  Values for each frequency band (in csv format):

 Frequency range (Hz), left channel value, right channe$
 9000-10000,0,0.000215
 8000-9000,0,0
 7000-8000,0.000314,1.7e-05
 6000-7000,0.00076,0.002116
 5000-6000,0.02367,0.031323
 4000-5000,0.065505,0.12749
 3000-4000,0.726508,0.776972
 2000-3000,0.315691,0.375329
 1000-2000,0.463413,0.51981
 0-1000,0.163606,0.215225

  Plot of proportions in each band:

   Left channel
    Freq. range (Hz)         |-------------------|
    9000-10000               |
    8000-9000                |
    7000-8000                |
    6000-7000                |
    5000-6000                |*
    4000-5000                |*
    3000-4000                |**************
    2000-3000                |******
    1000-2000                |*********
    0-1000                   |***

   Right channel
    Freq. range (Hz)         |-------------------|
    9000-10000               |
```

In order to calculate the Gini index, the packages *seewave*, *tuneR*, and *ineq* are required to be installed in R. Go to Packages and Install Packages. Scroll down and select *ineq*. Once *ineq* is installed, from the R menu, select Packages and Load, select the *ineq* script in R in the same way as the previous scripts.

The *ineq* inscript will provide the proportion per band and the Gini index, in the case of the night file (file1.wav):

```
Gini coefficient:

Left channel: 0.659785
```

The morning file (file2.wav) results are:
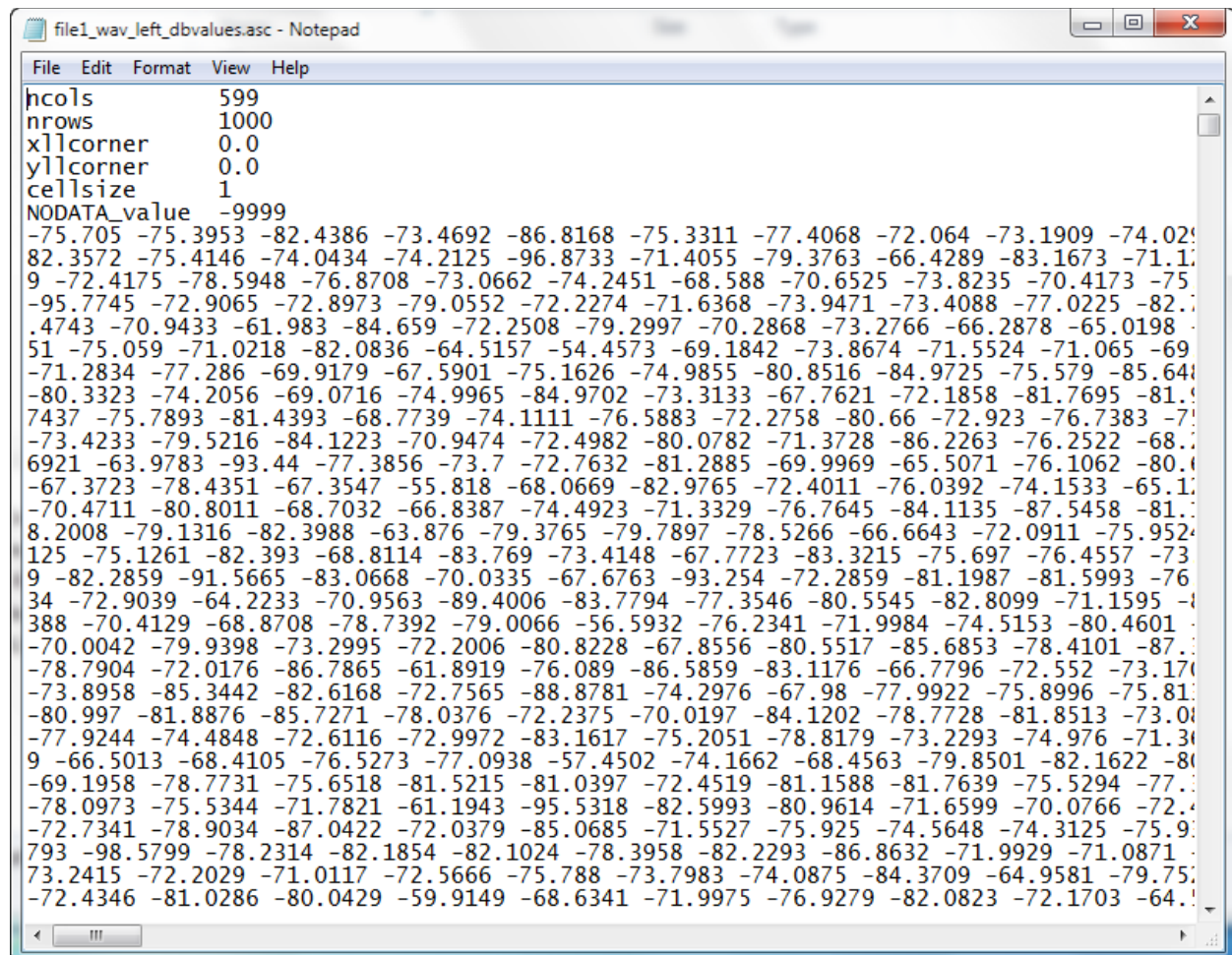
```
Gini coefficient:

Left channel: 0.473508
```

2. Batch file processing[2]. The batch file version of this script requires all three R packages (*seewave, tuneR, ineq*) to be loaded and the directory for all files to be processed to be selected using 'File' and 'Change dir' menu options. The output from this batch script is a file, called R_results.csv, containing the proportion of sounds occurring in each band for each recording, the Shannon's frequency band diversity value for each channel as well as the Gini coefficient for each channel. Each recording is organized as one record in the database like this (showing a plot of one day's worth of recording for the Purdue Wildlife Area):
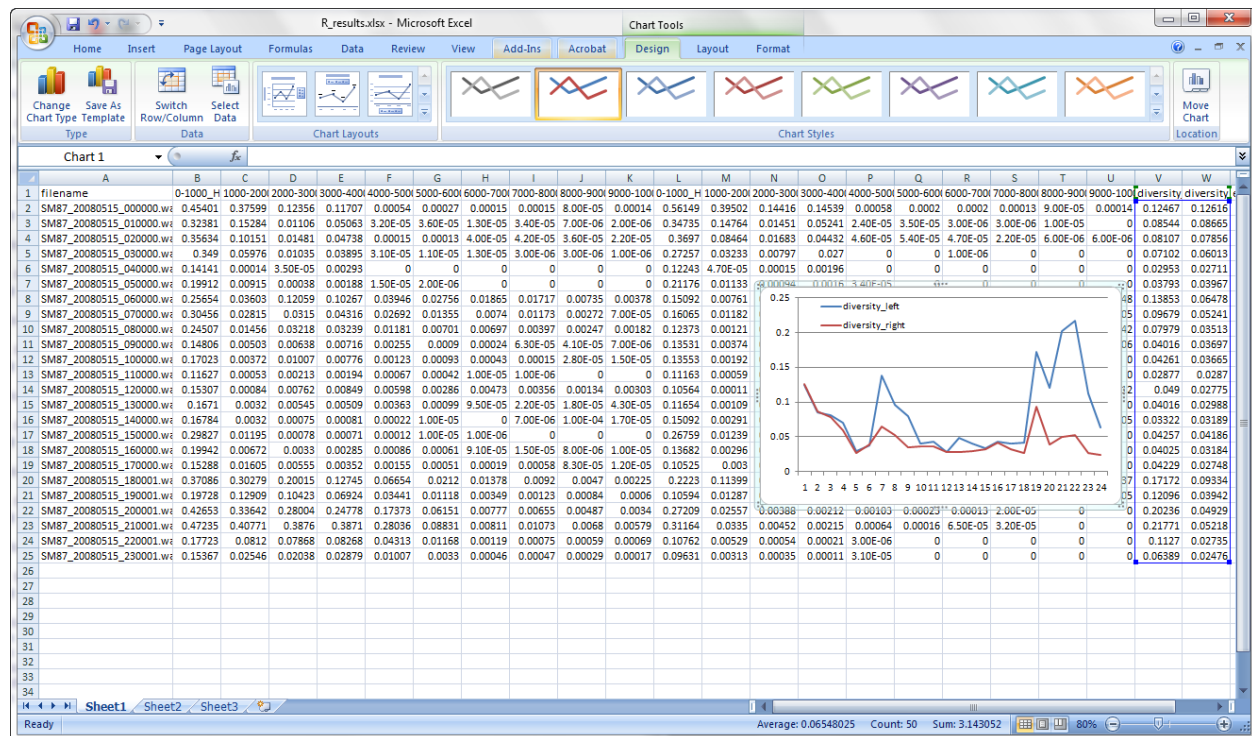
Both R scripts will output a raster version of a spectrogram that can be opened by any text editor. Opening the raster file associated with executing the single R script on file1.wave, the left channel ASCII representation of the spectrogram would show this:

Importing the ASCII file into ArcGIS 10.0 is straightforward for those users familiar with this GIS software package. Using ArcToolBox and then convert from ASCII to Raster
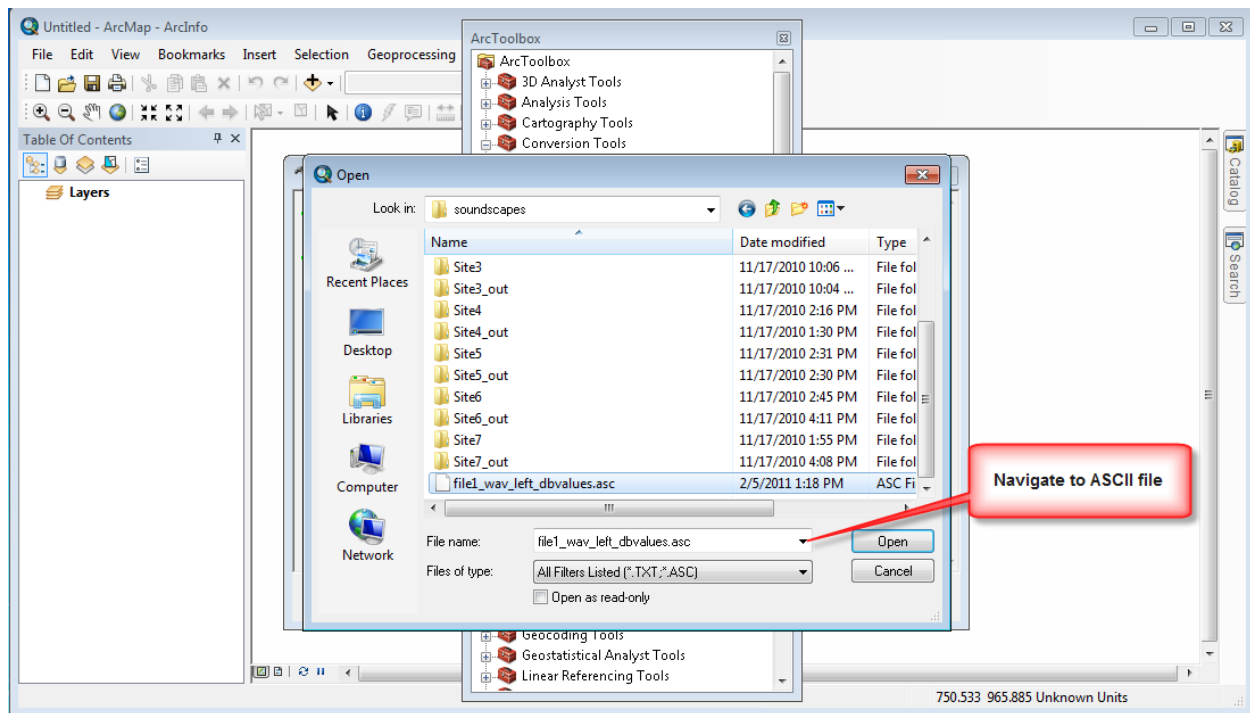
Since dBFS values are floating point, using either the FLOAT or INTEGER Output data type is possible.
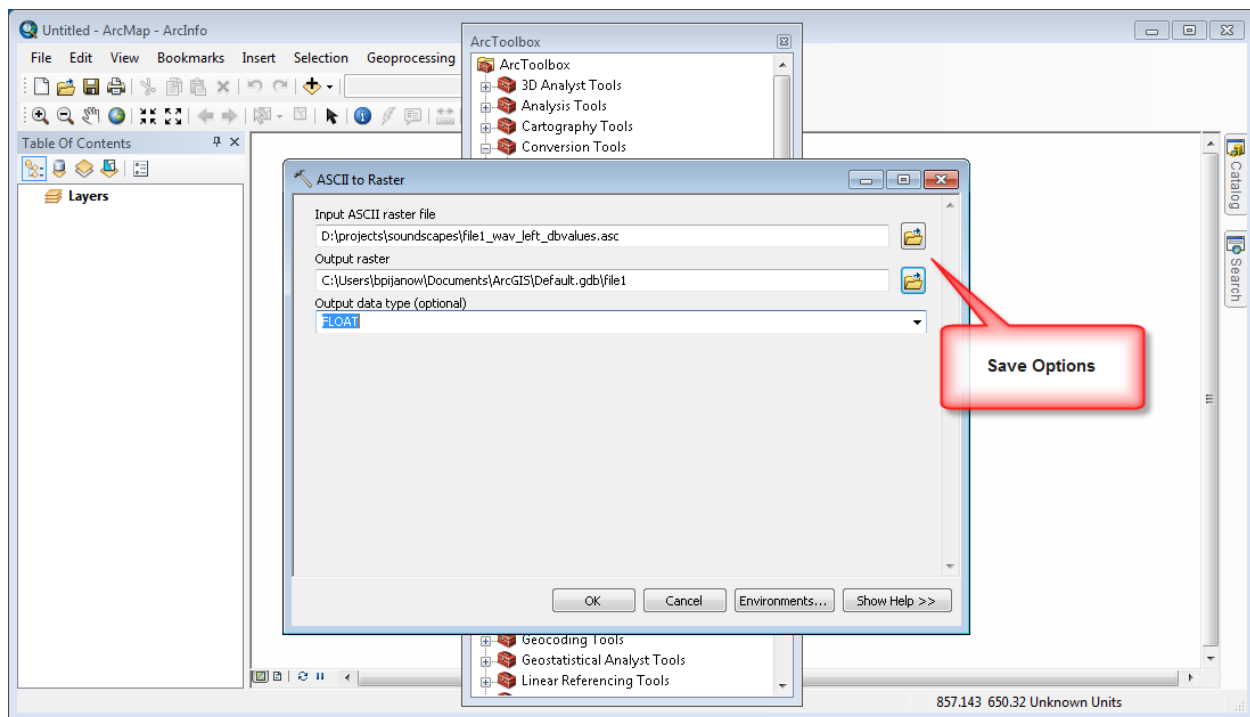
After the ASCII file is imported and the color ramp changed from the default grey scale, the file1.wav file for the left channel should look like a raster map:
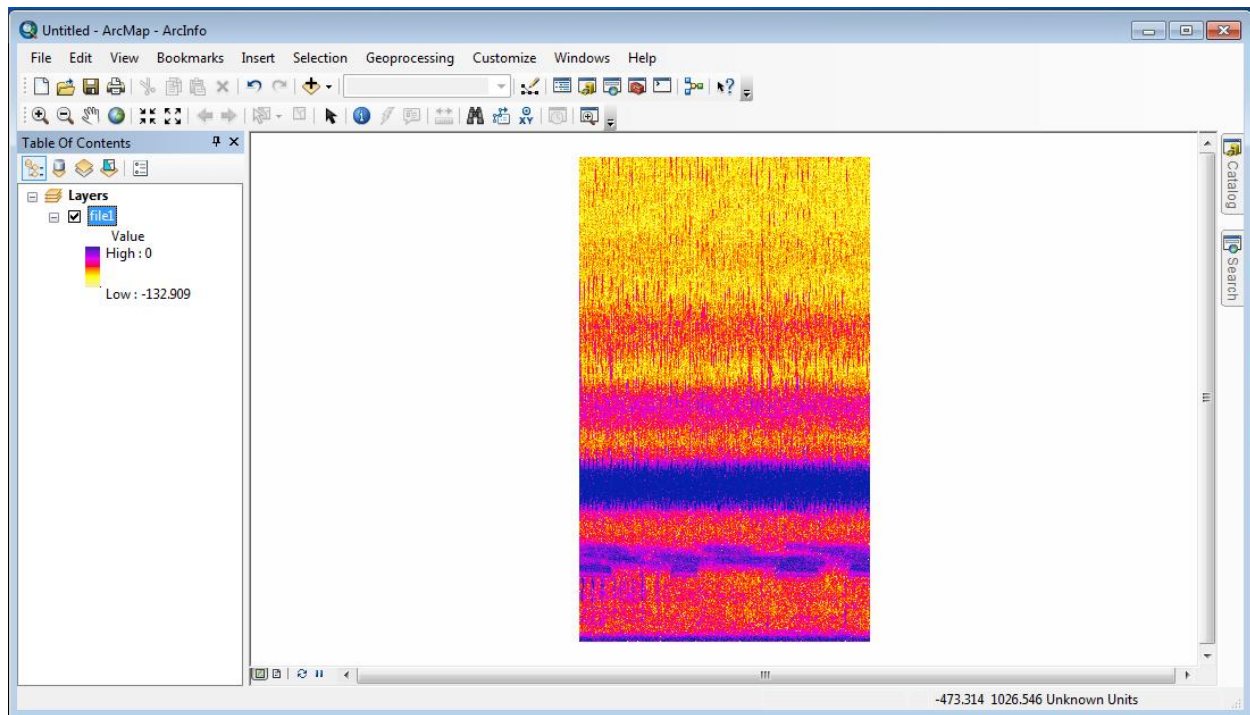
A zoom into the lower left shows the detail that is possible with this approach:
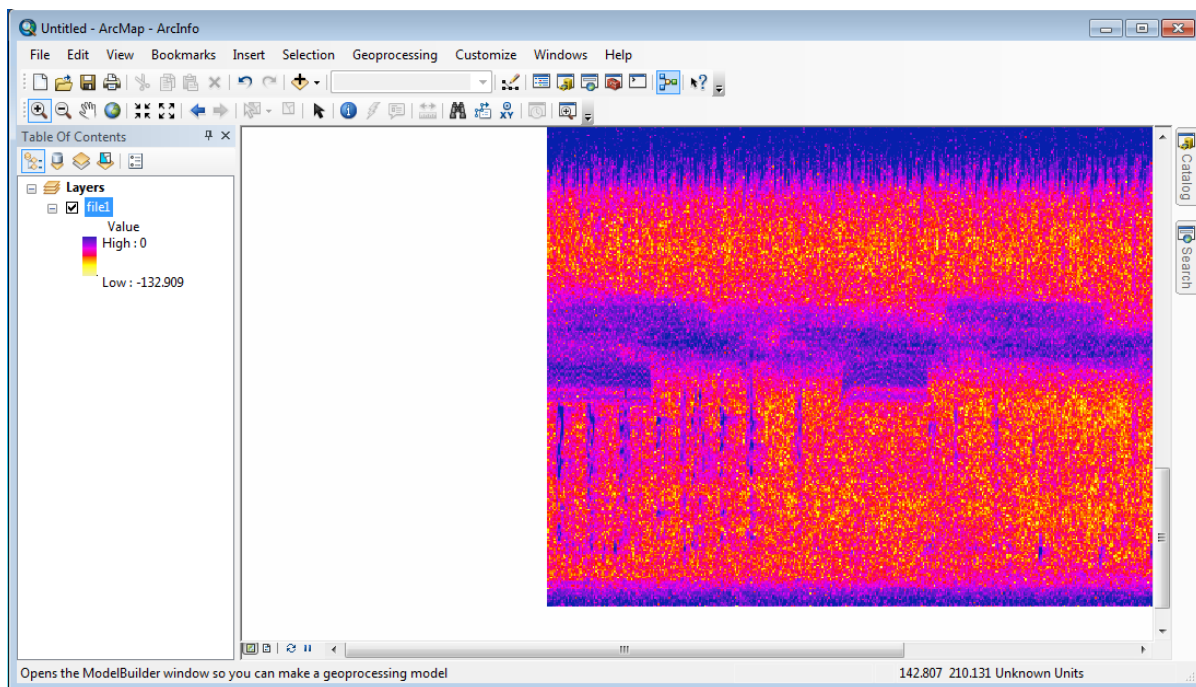
Using Raster Calculator in ArcGIS 10, those sounds that have an intensity greater than, viz. -50 dBFS, can be saved as a binary map. The equation in Raster Calculator is "file1" > -50
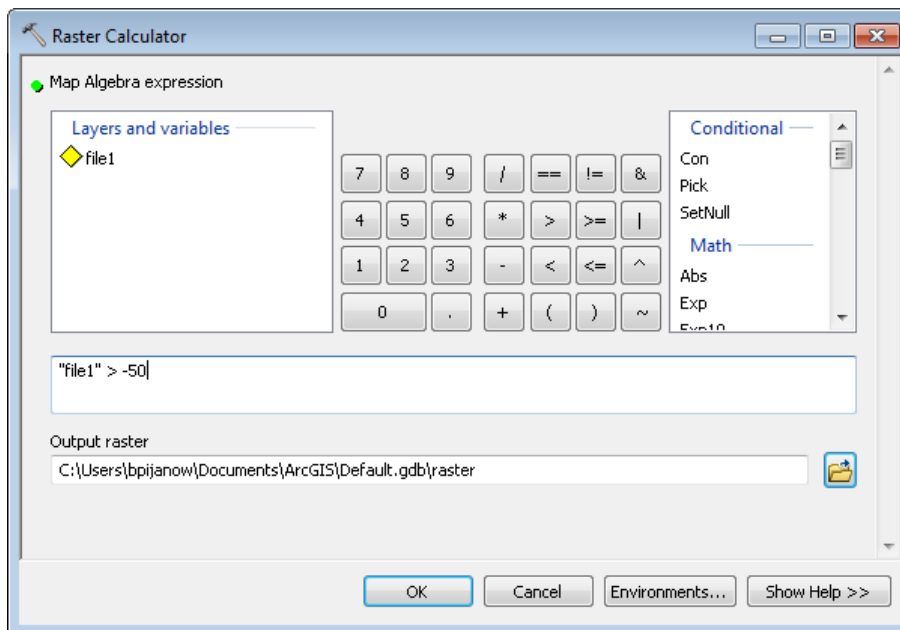
Hitting "OK" will produce the following binary map:
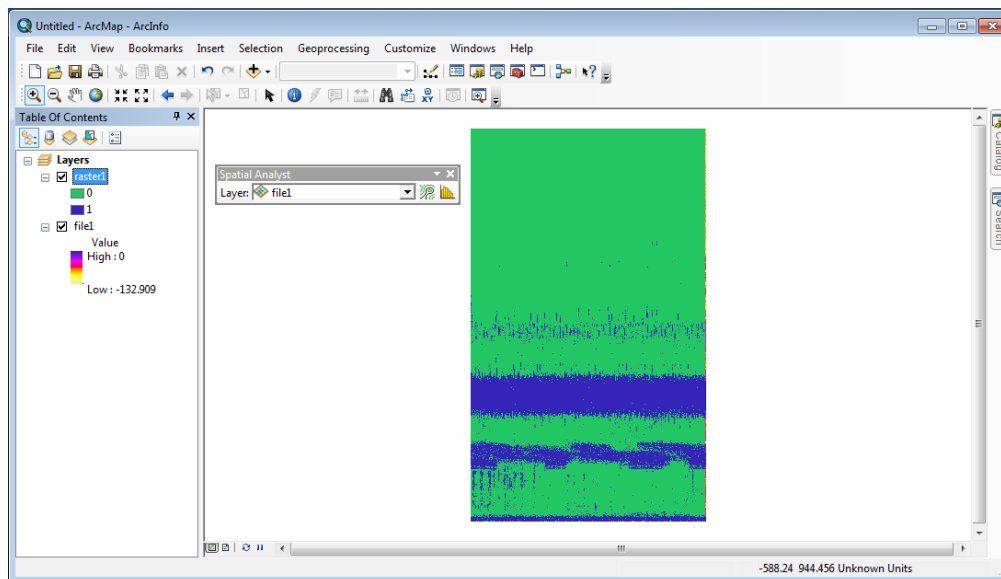
The python script that is covered in the next exercise performs this raster calculator step automatically. The python script does require the user to convert the ASCII files to an ArcGIS raster map.

EXPLANATION OF GINI COEFFICIENT

The Gini index computes the area under a Lorenz Curve where frequency bands are ranked from low occupancy to high and the area (A, light grey and dark grey areas) under this curve is compared to area contained with the condition where all bands have equal occupancy (B, dark grey area only), represented by the diagonal line. The ratio of A to B is the Gini coefficient.

Values of 1.0 are conditions where all bands have the same amount of sound and 0.0 is the condition where all sound that occurs does so in only one band. The Lorenz curve and the pure equality line (the situation of purely even classes) are shown here:
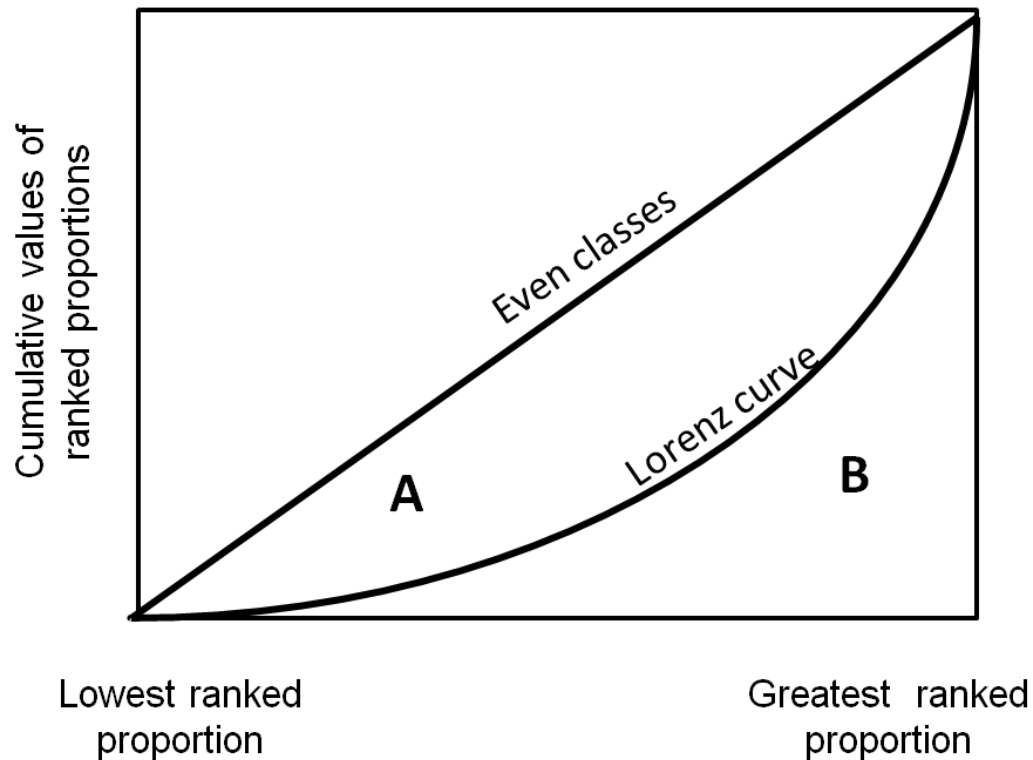
Taking the output from the single file R script on file1.wav which is input to Excel is shown next. A plot of the right and left channel proportion of sounds in each frequency band is shown as a bar chart. Using the values from the right channel, Gini is calculated using the following steps. First, values from the right channel are order from lowest to highest (shown as rank order). The cumulative value is then calculated at each level of the rank to produce values that make up the Lorenz curve (e.g., column labeled Lorenz curve):

band	left	right	rank order	Lorentz Curve
9000-10000	0.000	0.000	0	0
8000-9000	0.000	0.000	0	0
7000-8000	0.000	0.000	0	0
6000-7000	0.001	0.002	0.002	0.002
5000-6000	0.024	0.031	0.031	0.033
4000-5000	0.066	0.127	0.127	0.16
3000-4000	0.727	0.777	0.215	0.375
2000-3000	0.316	0.375	0.375	0.75
1000-2000	0.463	0.520	0.52	1.27
0-1000	0.164	0.215	0.777	4.047
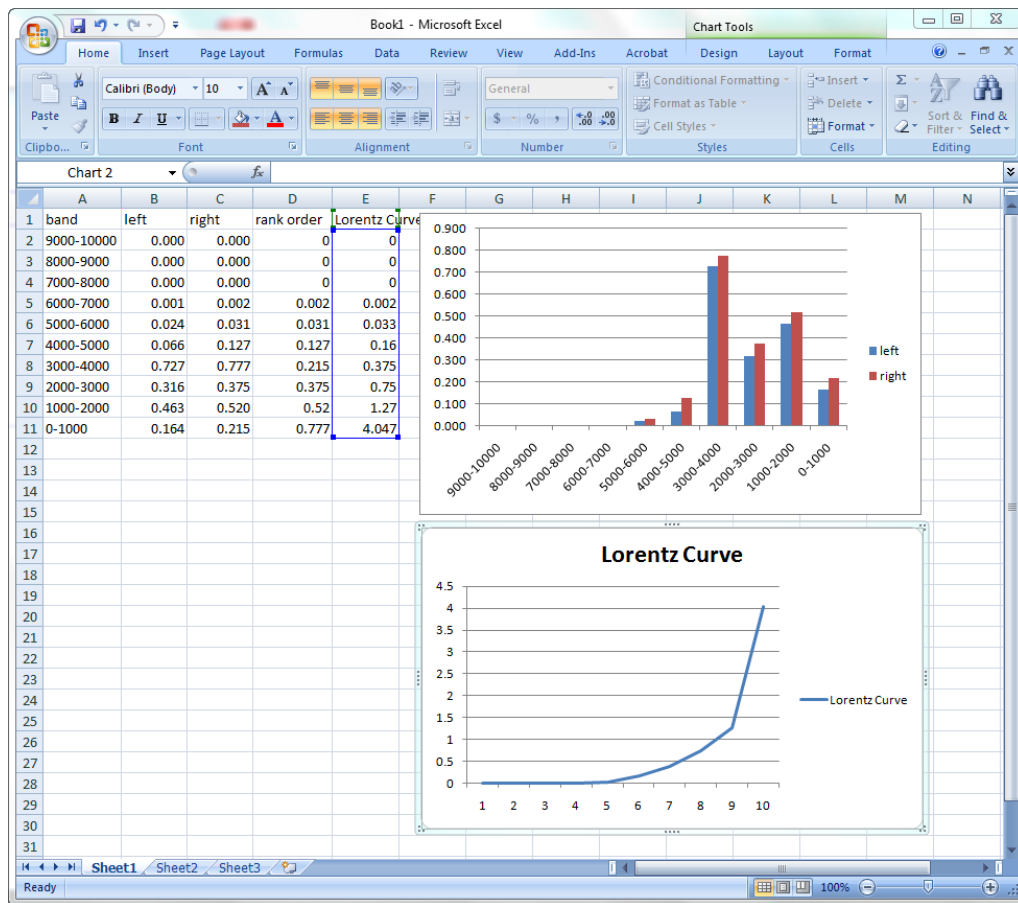
The area under the Lorenz curve (area B) is approximately by summing all values in this column. The area under A is calculated as ½ * max value * N where N= number of frequency bands. Gini is then A/B.

EXERCISE 3 – SPECTROGRAMS AS RASTER FILES
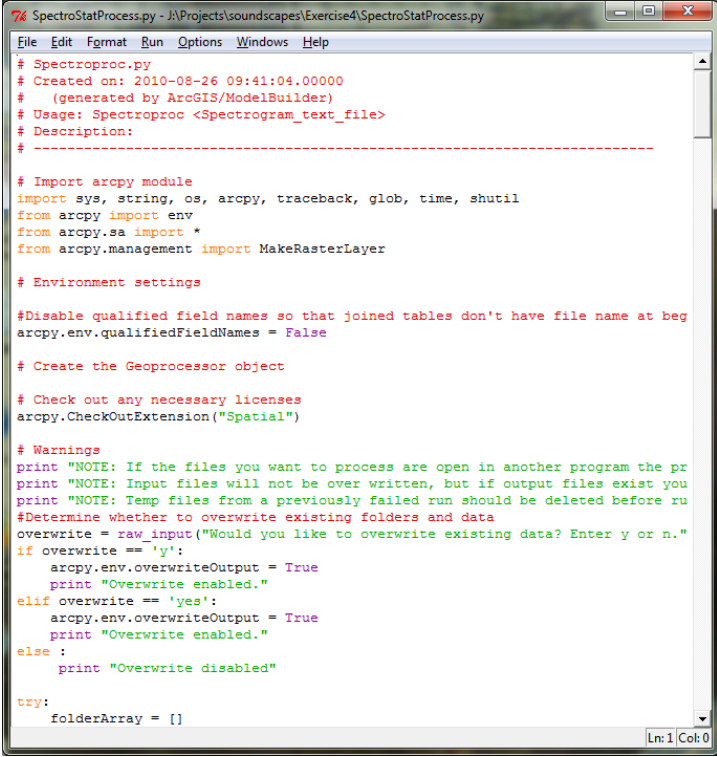
In the previous exercise *seewave* was used to export a spectrogram as an ASCII raster file. This exercise will take advantage of tools traditionally used for Geographic Information Systems (GIS) to analyze the spectrograms. Note that ArcGIS 10.0 and Python 2.6 programs are required for this exercise.

A python script was used to link several tools together and iterate through the steps required to complete the analysis. The script creates polygons *(i.e.* patches) of sound based on a user specified dBFS level, and then generates statistics for these polygons that describe the sound within. It is designed to process multiple days' worth of data from multiple sites, but for the purposes of this exercise we will examine a single file.

GETTING STARTED

The examples in this exercise will be completed using the integrated development environment (IDE) software installed by default with Python called IDLE. For different IDE options and to learn more about Python see the Python Wiki (wiki.python.org).

1. Begin by opening the script in IDLE. Browse to the folder location where the exercise data was saved, right click the SpectroStatProcess.py file, and choose "edit with IDLE".
2. This will open two windows. Leave both windows open as one is used to display the code while the other displays the process results.
3. In the window labeled SpectroStatProcess.py click the Run drop down menu and choose Run Module.

4. The program will run in the Python Shell window. It begins by writing several notes for processing issues encountered during more advanced runs., and then asks whether to overwrite existting data. Input data is never overwritten, and only output from previous runs will be overwritten by enter y. Enter the letter "y" and press the enter key.

5. The script is designed to process either one folder of data, or a folder containing several folders worth of data (e.g. each site in a different folder). For this exercise there is only one folder so choose "n".

6. Next enter the full directory path for the data. The easiest way to enter the path is to browse to the folder in windows explorer and copy the path from the address bar and paste it into the python window.
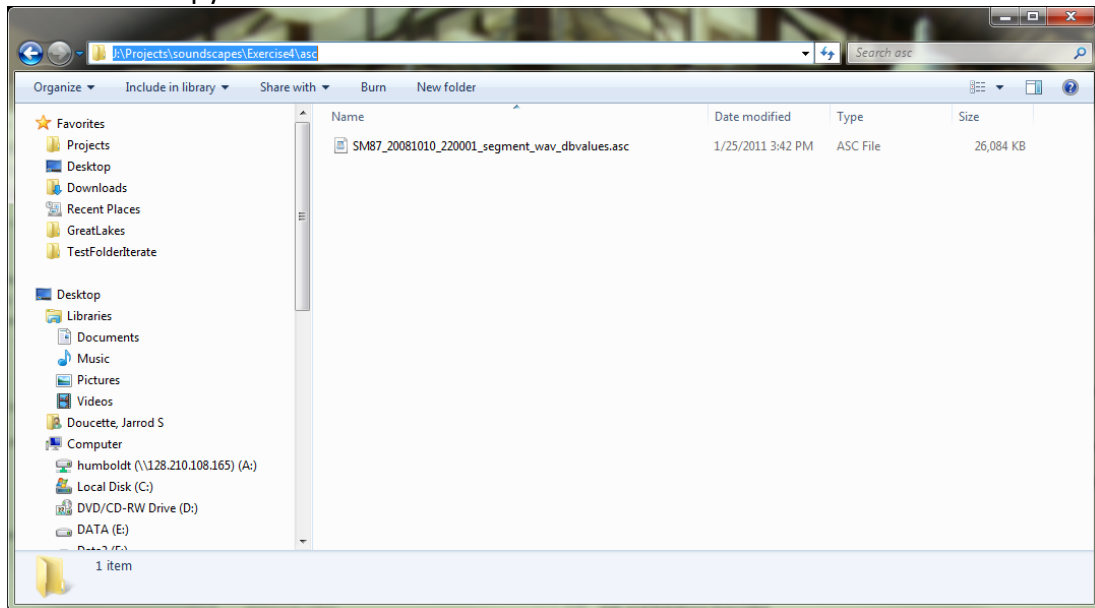
7. Now enter the values, in dBFS, to use as the cutoff for defining a patch of sound. In this case enter -50 and -60. The numbers entered are confirmed on the screen and the number of times each file will process.

8. The script is now running. There are many screen outputs designed to provide information about the current file being processed, and past processes that have run. When processing multiple files and folders worth of data it shows how many files and folders have been processed and the total number to be done. Time information such as when file processing started, how long each file took to process, and how much longer it is expected to run is also available. On a moderately fast computer, with a 3.1GHz processor and 4GB of memory, it typically takes 4-8 minutes per file depending on the complexity of the soundscape (i.e. number of polygons).

WHAT IS HAPPENING

1. The python script takes the ASCII files and creates a binary image based on the cutoff values from step 7 above.
2. Next the binary image is smoothed by running an 8 X 8 majority filter three times in order to reduce the number of polygons to <100,000 on average.
3. The raster file is then converted to a polygon shape file where each polygon represents a patch of sound.
4. Fields are added to the shapefile table to hold information about the file being processed and statistics describing the different sound patches.
5. Date and sensor information about the recording are extracted from the file name and added to fields (sensor, year, month, day, hour, row count, and column count)
6. Next information about the sound patches (area, perimeter, x, y coordinate of centroid) is calculated for each polygon.
7. Finally zonal statistics are run on the spectrogram for each polygon (min, max, range, mean, standard deviation, sum, variety, majority, minority, and median).
8. The resulting shapefile can be viewed in ArcGIS or the dbf portion of the file can accessed using spreadsheet software such as Excel.

ANALYZING THE OUTPUT

We can now look at the polygon file of sound patches and the associated statistics that were generated by the script. For this exercise use ArcGIS to visually examine the sound patches and look at their associated attributes.

1. Open ArcGIS and add the shapefile from the newly created asc_out folder.
2. The lower cutoff used for the -60 dBFS file will capture a larger amount of sound patches so it's best to place it beneath the -60 dBFS file for visualization.
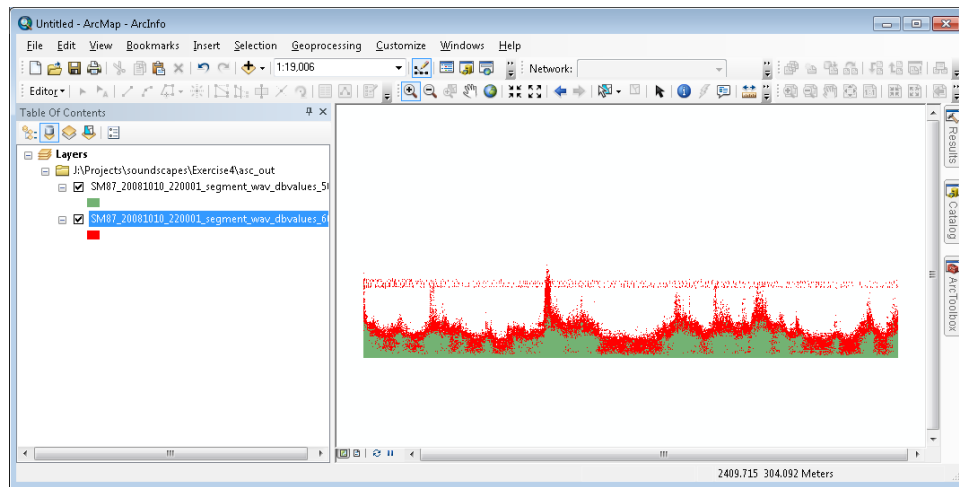
Notice that the -60 dBFS file (red) extends above the -50 dBFS file (green), and that it also picked up a series of repeating sounds in the upper frequencies. This pattern is the repeating calls of the frogs heard in the first exercise. The red peaks represent cars driving past the recorder.

3. The statistics produced by the script are useful for dissecting sounds within a spectrogram. Use the ID tool to click on one of the sound patches in the upper frequency range.
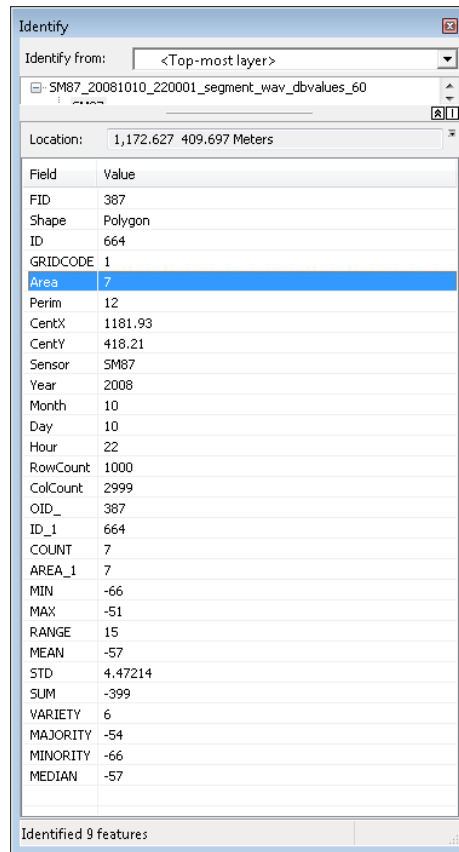
4. Begin by looking at the area field. This field represents the size of the sound patch measured in spectrogram pixel units. Pixels are one time unit (i.e. 1/10 of a second) wide and one frequency unit (*i.e.* 10 HZ) tall. In this case a small patch of 7 pixels has been identified.

5. Next the CentX field provides information on the time the sound occurred. Each pixel from the spectrogram represents 1/10 of a second. In this case the sound occurred at 1,181.93 tenths of a second, or 1.97 minutes into the recording.

6. The CentY field provides information on the frequency of the sound patch. Each pixel represents 10 Hz, which means the center of this sound patch occurred at approximately 4.2 khz.

7. The Sensor, Year, Month, Day, Hour fields were extracted from the file name by the script. These fields are most useful when comparing multiple recordings.

8. RowCount and ColCount are derived from the spectrogram raster during processing and are most useful for ensuring the frequency range and recording time are consistent when comparing multiple files.

9. The statistical measures beginning with MIN and ending with MEDIAN were computed based on the spectrogram pixels for each sound patch polygon.

    a. Notice the mean value of -57 dBFS is just above the cutoff value of -60 dBFS. Choosing a lower cutoff of would create large sound patch, but choosing too large of a cutoff value can cause patches to expand into one another and cause loss of detail.

---

b. Also notice that the minimum dBFS value in the patch is -66 dBFS, which is below the cutoff of -60 dBFS. This is a result of the smoothing process (i.e. majority filter tool) used to reduce noise within the spectrogram.

CONCLUSION

The script analyzes spectrogram files in units of sound patches. The attributes generated provide information about the time and frequency for each sound patch as well as statistics describing the intensity of the sound. Varying the cutoff frequencies results in outputs that capture different phenomenon and the script is capable of generating multiple output files at once making it easier to find a suitable sound patch file.

You should now be able to process spectrograms to examine sound patches. You can download more recordings from www.purdue.edu/soundscapes and process several sites or days of data.

REFERENCES

Charif RA, Ponirakis DW, Krein TP. 2006. Raven Lite 1.0 User's Guide. Cornell Laboratory of

Ornithology, Ithaca, NY.

Ligges U. 2009. tuneR: Analysis of music. http://r-forge.r-project.org/projects/tuner/.

Pijanowski, B.C., A. Farina, S Dumyahn, B. Krause, and S. Gage. 2011. What is soundscape

ecology? Landscape Ecology 26: 1213-1232. doi: 10.1007/s10980-011-9600-8

Sueur J, Aubin T, Simonis C. 2008. Seewave: a free modular tool for sound analysis and

synthesis. Bioacoustics 18: 213-226.

Villanueva-Rivera L, B.C. Pijanowski, J. Doucette, and B. Pekin. 2011. A Primer of Acoustic

Analysis for Landscape Ecologists. Landscape Ecology 26: 1233-1246. doi:

10.1007/s10980-011-9636-9