VSRT Memo # 054

Script for simple MOSAIC data analysis using GNU Octave

Vincent L. Fish

Abstract

The free GNU Octave package can be used to reduce MOSAIC data in an elegant way using matrices. Data reduction with Octave may represent an alternative analysis pathway with pedagogical value reinforcing linear algebra concepts for students of an appropriate mathematical level. A sample, annotated Octave script is provided.

1 About GNU Octave

GNU Octave is a free software package intended for numerical computations that is mostly compatible with MATLAB, a commercially-available product of The Mathworks. Full details of Octave, including a manual and links to download the package, are available at `http://www.gnu.org/software/octave/` , and additional useful packages can be found at `http://octave.sourceforge.net/` . Binary packages are available for Windows, Linux, and Mac OS X. The testing described in this memo was done on a Linux Debian-based system.

Like MATLAB, Octave is designed for fast performance on matrices. Octave supports coding with multiple `for` loops, as are common in any other programming language. However, faster performance and cleaner code can be obtained by using matrix representations of data. The remainder of this memo demonstrates some of the matrix-based data analysis that can be done with Octave.

2 Data format

The input file `chsout.txt` consists of lines of the form

```
year 2008 day 026 hour 00 min 00    num_rec   6 sun_el -25.1
 tpwr 116.12705 rms 0.06847 ltm   20.1 spectrum
```

(not including the line wrap) followed by a series of 64 numbers corresponding to data in each of the spectral channels. The downloaded file contained 45712 lines. The number of lines can be obtained with the Unix command `wc` or inferred later in Octave. (It is not necessary to know this number ahead of time.)

3 Script

Octave will read scripts in the current directory. Scripts should have an extension `.m` and can be invoked by typing the filename (without `.m`). Thus, if your script is called `mosaic.m`, simply type `mosaic` at the Octave prompt.

A sample script appears below, with annotations following. The four lines beginning with `[data,count]` should appear on one line but have been wrapped for display here. Lines beginning with `#` are interpreted by Octave as comments. It is not necessary to terminate commands with a semicolon, but not doing so will cause Octave to echo the result of the calculation.

```
# MOSAIC - A script file to load and do basic analysis of MOSAIC data
# -----------

# Read in data into a 73 row x 45712 column matrix
fid=fopen('chsout.txt','r');
# The following prints as four separate lines but should appear on one
[data,count] = fscanf(fid,"%*s %f %*s %f %*s %f %*s %f %*s %f %*s %f %*s %f
%*s %f %*s %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f
%f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f
%f %f %f %f %f %f %f %f %f %f %f %f %f",[73,Inf]);
fclose(fid);

# Figure out how many lines of data we read in
n = count/73;

# These are 1 row x 45712 column matrices
year = data(1,:);
day = data(2,:);
hour = data(3,:)
minute = data(4,:);
num_rec = data(5,:);
sun_el = data(6,:);
tpwr = data(7,:);
rms = data(8,:);
ltm = data(9,:);

# This is a 64 row x 45712 column matrix
spectrum = data([10:73],:);

# This is a 45712 row x 1 column matrix
unitcolumn = ones(n,1);

# To sum a quantity, right multiply by unitcolumn, e.g.:
averaged_spectrum = spectrum*unitcolumn/n;
#plot(averaged_spectrum);

total_records = num_rec*unitcolumn;
```

```
# Note that we’re conjugating num_rec
weighted_spectrum = spectrum*num_rec’/total_records;
#plot(weighted_spectrum);

# Take only scans where the sun is below -30 deg elevation:
# k has dimension 1 x 45712
k = sun_el < -30;
left_vector = ones(64,1);
mask = left_vector*k;

# Note the . below -- doing element-by-element multiplication
#   NOT MATRIX MULTIPLICATION!
masked_spectrum = spectrum.*mask;

total_masked_records = k*num_rec’;

weighted_masked_spectrum = masked_spectrum*num_rec’/total_masked_records;
plot(weighted_masked_spectrum);
axis([1 64]);
xlabel(’Channel number’);
ylabel(’Uncalibrated units’);
legend(’off’);
```

4 Explanation of the script

The first three commands open the data file for reading, load the data into a matrix, and close the file. The Octave `fscanf` command is similar to the C command but is less fussy about number type and precision. In the format string, `%*s` tells Octave to read in and subsequently ignore a string, and `%f` tells Octave to accept a number. The `[73,Inf]` asks Octave to construct a matrix called `data` consisting of 73 rows (corresponding to the 73 numbers read in) and as many columns as necessary. The variable `count` contains a count of the number of successful numbers read in (73 × 45712).

The matrix `data` looks like

$$\begin{bmatrix} 2008 & 2008 & 2008 & 2008 & \dots \\ 26 & 26 & 26 & 26 & \dots \\ 0 & 0 & 0 & 0 & \dots \\ 0 & 10 & 20 & 30 & \dots \\ 6 & 7 & 6 & 7 & \dots \\ -25.1 & -27.1 & -29.1 & -31.1 & \dots \\ 116.12705 & 116.13938 & 116.13049 & 116.12940 & \dots \\ 0.06847 & 0.07741 & 0.07153 & 0.06976 & \dots \\ 20.1 & 20.3 & 20.4 & 20.6 & \dots \\ -0.06246 & -0.00345 & -0.02799 & -0.09357 & \dots \\ 0.02521 & -0.00580 & 0.04384 & 0.10648 & \dots \\ 0.00073 & -0.02041 & -0.02089 & 0.05103 & \dots \\ \dots & \dots & \dots & \dots & \dots \end{bmatrix},$$

where each column consists of one data point (45712 in total) and each row consists of a series of 73 numbers corresponding to year, day, hour, etc., ending in 64 numbers representing the spectrum. The next lines in the script split out the first nine rows into row vectors of their own and rows 10 through 73 as a 64 × 45712 matrix (called `spectrum`) of the spectrometer data alone.

We can produce a summed spectrum by adding up the points of each row. The simplest way to do this is to right-multiply the matrix `spectrum` by a 45712 row × 1 column vector consisting entirely of ones. The script demonstrates how to set up a unit column vector and do matrix multiplication. The same line also divides every element in the product matrix `spectrum*unitcolumn` by the scalar `n`.

However, a fairer way to average the data is to weight the data points by the number of records that have been pre-averaged to produce the 10-minute data record. In other words, we want to find

$$\bar{d}_i = \frac{\sum\limits_{j=1}^{n} d_{i,j}\, w_j}{\sum\limits_{j=1}^{n} w_j},$$

where j ranges from 1 to the number of data points n, $d_{i,j}$ is the jth data point in spectral channel i, and w_j is the number of records associated with the data point. As before, the sum in the denominator can be computed by right multiplication of the 1×45712 matrix `num_rec` with the 45712×1 matrix `unitcolumn`. The sum in the numerator can be obtained by multiplication of the data points by the number of records and then summing them. However, `spectrum` is a 64 × 45712 matrix and `num_rec` is a 1 × 45712 matrix, so they cannot be multiplied directly. Instead we must take the transpose of `num_rec` and form a 45712 × 1 matrix. To transpose a matrix in Octave, add `'` after the name.

Finally, we can select subsets of the data by producing a mask vector (or matrix) with ones representing the data we wish to keep and zeroes representing the data we wish to discard. For instance, to select only data for which the elevation of the Sun is less than −30°, the command is `k = sun_el < -30`. Since `sun_el` is

$$\left[\begin{array}{cccccccc} -25.1 & -27.1 & -29.1 & -31.1 & -33.1 & -35.0 & -37.4 & \ldots \end{array}\right],$$

the vector `k` will have the following entries:

$$\left[\begin{array}{cccccccc} 0 & 0 & 0 & 1 & 1 & 1 & 1 & \ldots \end{array}\right].$$

Left multiplication by a 64 × 1 unit vector will produce a 64 × 45712 matrix (`mask`) consisting of 64 copies of the 1 × 45712 row vector `k`:

$$\left[\begin{array}{cccccccc} 0 & 0 & 0 & 1 & 1 & 1 & 1 & \ldots \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & \ldots \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & \ldots \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \end{array}\right].$$

We can now produce a new matrix corresponding to `spectrum` where data from Sun elevations greater than $-30°$ are replaced with zeroes by doing element-by-element multiplication of the 64 × 45712 matrix `spectrum` with the 64 × 45712 matrix `mask`. Careful, this is not matrix multiplication! The element-by-element multiplication operator in Octave is `.*` (note the period before the asterisk). To do a proper weighted average of the data, we must also zero out the weights from times when the Sun is above $-30°$ elevation. (While $d_{i,j}\, w_j$ will be zero for these data points, w_j will not, so the denominator in the equation for $\bar{d}_i$ will be incorrect if the high-elevation w_j terms are not set to zero.) The spectrum thus obtained is plotted in Figure 1. Multiple such masks can be constructed based on arbitrary filtering criteria, and the data can easily be filtered by doing element-by-element multiplication on all such matrices.

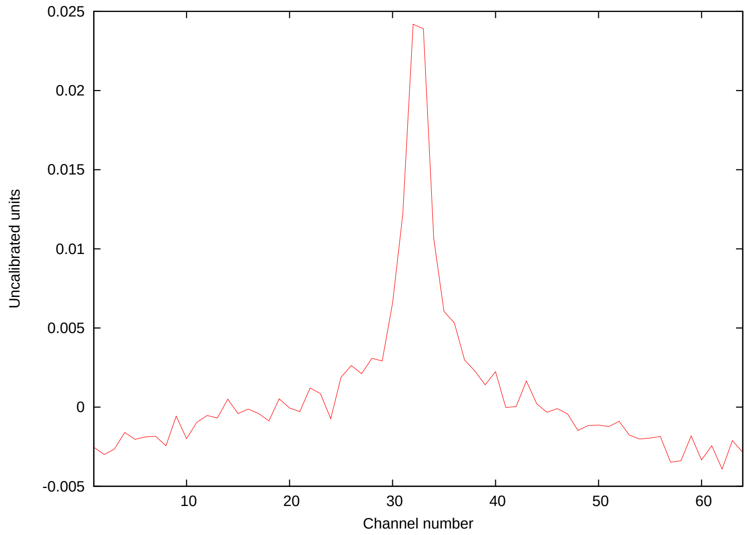

Figure 1: Spectrum of MOSAIC data for Sun elevations less than $-30°$.