# Lab 1 - Part 2: Environmental Data Sets

## *Lab Overview*

This is the second part of Lab 1; it's purpose is to show you some ways to import real data sets into MATLAB. Real data sets are often large, come in a variety of formats, and can be finicky to deal with. Therefore, it may take you a few tries to get this working: be creative and persistent!

## *Learning Goals*

After this lab you should be able to :

- Load a variety of real data sets into MATLAB
- Create and use a mask to remove missing values from data
- Create a vector of datenumbers using a for-loop
- Generate x-y plots of loaded data with appropriate axes and figure captions

## *To Hand In*

1. A plot of the Mauna Loa CO2 time series.
2. A plot of the MEI index.
3. A plot of the data you downloaded for the pre-lab.
4. A figure caption for the plot of your data set.

## 1. Mauna Loa $CO_2$ time series

First you will make your own plot of the Mauna Loa data which you saw on the first day of class.

Before you begin, remember to create a new sub-directory (folder) in your Z: drive where you will be able to save all files from this lab session. Ensure MATLAB is working in this directory.

Download the file `monthly_maunaloa_co2.csv` from the class website on Connect. This contains atmospheric $CO_2$ data recorded at the Mauna Loa Observatory. Save the file in the correct directory.

Now open the file in a text editor[1] and observe how the data is organized. Notice the detailed header; we'll have to tell MATLAB to ignore the header, but you should at least scan it so you know what the data represents. How many header lines are there? What do the columns represent?

First, we are going to load the data from the file into MATLAB with the `textscan()` function. This function uses the following input arguments:

- a file identifier created by using the function `fopen()`
- a string specifying the format of the data. Use `%f` for a float, `%d` for an integer, and `%s` to specify a string. For clarity, we will create a separate variable containing this information.

as well as the following *options*

[1] The default *TextEdit* isn't great for this task, and neither is *MS Word*. Instead, try using *Vim*. Right click the file, select *Open With* and then click on *Vim*. You can also see the file directly in MATLAB by entering `> edit your_filename` at the command prompt.

- the number of header lines
- the delimiter separating columns, in this case a comma.

```
fid = fopen('Your_filename_here.csv');
format = '%f %f %f %f %f %f %f %f %f %f';
mauna_loa_data = textscan(fid, format, 'HeaderLines', 57, ...
     'Delimiter', ',');
```

This snippet of code will create the *cell array* 'mauna_loa_data' containing ten arrays, one for each column in the csv file[2]. We haven't talked about cell arrays yet; a cell array is an array (a vector) of cells where each cell can contain any object you specify. In this case, each cell contains a vector of floating point numbers which represents one column of your original data. However, we are going to take the contents of the cells we are interested in and assign these to their own variables. We want cells 4 and 5, containing the date and measured $CO_2$ . Curly brackets are used to access a cell within a cell array:

```
co2_date = mauna_loa_data{4};
co2 = mauna_loa_data{5};
```

The easiest way to look at this data is to plot it:

```
plot(co2_date, co2);
```

Notice the negative values in the time series. These are missing values. Go back to the header, determine how missing values are indicated, and remove these from the data set using the masking technique you learned in Part 1 of this lab. Then make a new plot, adding the correct labels on the x- and y-axes. Remember that you can use the `axis()` command to play around with the axis limits.

Save the plot to a PDF or JPG file. Please print and submit this figure.

## 2. MEI Index

Now we are going to import and plot a time series of the Multivariate ENSO Index (MEI), a climate index which represents the state of the El Niño-Southern Oscillation. The MEI gives an indication of whether the tropical Pacific is in an El Niño (warm water, positive MEI) or a La Niña (cold water, negative MEI) state.

From the class website, download the file `MEI.html` and place it in your working directory. As before, open the file in a text editor and observe how the raw data is organized. This is always your first step.

Notice, there are 13 header lines, 63 rows of data, all followed by some user tips at the end of the file. Each row represents a year and each column a month within that year.

Though we could again use `textscan()` to upload the data, we will use a different approach this time. In the future, feel free to use whichever suits you better.

[2] The three dots at the end of the second-to-last line tell MATLAB that the code continues on the next line

Begin by making a copy of the file and renaming it something like `MEI_data_only.txt`. Now, in this new file, simply delete *all* the header lines as well as the trailing information, leaving you with *only* the 63 lines of data. Make sure to save.

Notice that the last line of data doesn't contain the same number of columns as all the others. This is because we don't have data from Aug-Dec 2012. However, when we use the `load` command (below) missing data cannot be indicated with blank spaces because, as we've seen before, MATLAB ignores white space. Therefore, manually enter the five missing data points as `NaN` which simply stands for 'not-a-number'. The final line in your data file should now look like this:

```
2012  -1.046  -.702  -.41  .059 .706 .903 1.139 NaN NaN NaN NaN NaN
```

Don't worry if the numbers aren't evenly spaced. Remember, excess white space is ignored.

Now with a text file containing only data and no extra information, it's a breeze to load it into MATLAB:

```
M = load('MEI_data_only.txt');
```

With this, `M` is a matrix containing all the data from the text file. Have a look at it in the *Workspace* or the *Command Window* to see how it is organized.

This complete, there are two challenges left before we can plot this data. First, we need to reshape the two-dimensional time series into a one-dimensional vector (call this `mei`) so that it can be plotted. Secondly, since our data only has year information but no month information, we need to create an appropriate vector (called `T`) the same size as `mei` to represent time, both year and month. The first step is quite easy as MATLAB has a built-in function to reshape matrices

```
mei = reshape(N', [len 1]);
```

Here, `N` is the matrix `M(:, 2:end)`. We do this because the first column of `M` doesn't contain any data. The variable `len` is an integer that you have to specify; it is the number of data points, or the number of elements in the matrix `N`. You'll need to define both `N` and `len` before the above line will run.

Now that you have a single one-dimensional array containing your data points, you need to make an array holding the corresponding dates. To do this, you can use the method outlined below. Once you've done this, make a plot of the MEI index (remember to label your axes) and submit this figure.

***Dates in MATLAB***

In order to make a proper time series in MATLAB, you have to understand a little about how the program stores and displays dates.

There are three types of recognized date formats in MATLAB. These are *date numbers*, *date vectors*, and *date strings*. To illustrate these, try the following at the Command prompt:

```
> date
```

This call returns today's date formatted as a character string. This is convenient to read, but not very useful for calculations or for plotting since a computer can only plot numbers, not words. Now try entering

```
> datevec(date)
```

This takes the string stored in `date` and turns it into a vector whose elements represent `[yyyy mm dd HH MM SS]`. This is useful for calculations and logical comparisons as you access the year, month, day, etc..., information separately. Now try entering

```
> datenum(date)
```

This should return something like `735145`. This is a date number; it represents the number of days since the year zero, and *this is the format you need to use for plotting*. Hours and minutes in a day are handled as fractions, so that the call `datenum([2013 1 11 6 0 0])` would return the number `735145.25` since at 6:00 am you've passed a quarter of the day. If you want to see an example of how this works when plotting, try this at the Command prompt:

```
> today = datenum(date)
> exm_dates = today : 0.5 : (today+10)  % increments of 1/2 day
> exm_data = sin(0:20)
> plot(exm_dates, exm_data)
```

Once you have your plot, keep it open, and enter the following command at the prompt

```
> datetick
```

and observe what happens to the x-axis on your plot. This is how a time series figure is created in MATLAB.

Now, say you are facing a problem where you have to create an array holding the correct date *numbers* for a time series. The solution is to use a nested for-loop to create an array of date *vectors* and convert these date vectors to date numbers. For example, if you have quarterly data (every 3 months, beginning in January) for the years 2011 and 2012, and you need to create the appropriate time array, you could enter something like this into your script

```
my_datevectors = [];
row = 0;
for year=2011:2012
     for month=1:3:12
          row = row + 1;
          my_datevectors(row, :)=[year month 1 0 0 0];
     end
end
```

Make sure you understand what is happening in this loop. Then have a look at the array `my_datevectors` in the Command window or in your Workspace. If you want to transform these dates vectors into date numbers, you can use something like

```
my_datenumbers = datenum(my_datevectors);
```

Have a look at your new array in the Command window or the Workspace. You would now be able to make a plot using this array.

## 3. Import and Plot Your Data Set

Import the data set you downloaded in the prelab and generate a plot from it.

Since each of the data sets is organized differently, it is up to you to determine how to properly import it. However, feel free to show your data set to a lab TA and discuss challenges associated with importing the different data sets - some data sets are much easier to work with than others are.

Write an m-file which will do the following:

- Load the data set you selected in the pre-lab assignment.
- Assign the data to appropriate MATLAB variables.
- Generate a plot of your data with any missing values removed. Give your plot an appropriate title and label your axes, including the correct units. If your data file contains more than one kind of measurement, choose only one variable to plot.

Once the plot is completed, write a caption for the plot. The caption should include:

- A title for the graph
- A description of what data the plot is showing
- Axes labels including the units of the data
- Any information necessary for interpreting the plot correctly.

Captions should be between 15 - 100 words long; the more complicated a plot is, the more description will be needed in the caption.

For example, a figure caption for the first plot we created might look like this:

Figure 1: Atmospheric $CO_2$. $CO_2$ concentrations in parts per million (ppm) recorded at the Mauna Loa Observatory from 1958 - 2011.