Basic Matlab Tutorial 2: Working with data

Getting Started

1. First, you need to get a copy of the data for the labs. Create a directory on your hard drive where you will store your data. I will assume you've called in "Omaha." Open a web browser and navigate to the "Lectures" page and click on Tutorial2. This will bring up a simple web page with the data and documents for this lab. Download the omaha.xls file and save it in your Omaha directory.

Preparing data for Matlab

2. Open omaha.xls in Excel. If you don't have access to Excel, please read through this section and then download "omaha.txt" from the website.

3. In order to get the data in omaha.xls into Matlab, we need to convert it to ASCII text and make sure it is arranged like a matrix. There are three problems we will have to deal with: non-numeric header information, non numeric row labels, and missing data.

4. There are four header lines, 3 with text and one blank. This information is useful (it tells where the data came from) even though Matlab won't like it. We need a way to keep the headers in the file but tell Matlab to ignore them. Matlab will ignore any line in a text file that starts with "%." So, put a "%" at the start of the first four rows (be sure to use column 1). You should also put a "%" in front of the blank line below the years.

5. In the first column there are a series of letters indicating the start of a particular month. We know each row is a week, so this information is easy to reconstruct—let's delete it. Select rows 7-58 of column 1 (everything to the right of the data) and choose "Delete" from the "Edit" menu. When prompted, allow cells to shift left. The year labels are now offset, so delete the blank cell to the left of the first year.

6. There are a few weeks for which no data exists. There are a couple of ways to deal with this. A traditional way is to replace blanks with some number outside the data range, say –999. This method requires you to carefully exclude the –999's from any calculations inside Matlab. A better option is to replace the blanks with "nan." Nan stands for "not a number" and it is an official value in Matlab (it's actually defined for any system implementing IEEE arithmetic). The nice thing about nan's is that Matlab's graphics routines ignore them. Enter "nan" in place of any blank cells (look in the first and last columns).

7. We're now ready to save. Select "Save As" and choose "Text (Tab Deliminated)" and call the file omaha.txt. (Excel will give you lots of warnings about formatting—ignore them).

Loading into Matlab

8. Start Matlab.

9. Now we're ready to work with some data. Like UNIX and DOS, the command window has a "working directory" associated with it. In other words, if you try to load an external file or execute a function, Matlab will look in the working directory first. To see your current working directory, type "pwd" (short for "print working directory"). This is not where you want to be. You need for your working directory to be the same location where you saved your data (Omaha). To change your working directory, type "cd newdirectory" where new directory is the name of the directory you want to go into. On Mac/UNIX, you would enter something like

```
>>cd /usr/myname/Omaha
```

   while on a PC you would enter something like

```
>>cd C:/UserFiles/myname/Omaha
```

10. Another way of changing your directory is to click on the "…" button at the top of the command window. This will bring up a file browser that you can use to locate Omaha.

11. Type "ls" and you should see omaha.txt. To load the data into Matlab, type "load omaha.txt". If the format of the file is not to Matlab's liking, it will complain, and you will have to go back to Excel and mess around. If you're sick of Excel, download omaha.txt from the website and use it.

12. After loading the .txt file, type "whos." Matlab should tell you that you have a variable called "omaha" that is 53-by-20. Type omaha(1:5,1:5) to see the first five columns of the first five rows. Are these the same numbers as in the Excel file?

13. We have some array manipulation to do before we can plot. What we're trying to do is create two column vectors, one with the time (in years) and the other with the data. The first thing to do, is to separate the years (row one) from the data. Type

```
>>year=omaha(1,: );
```

   to save the years into a variable called years. We can then delete it from omaha in one of two ways. The first way is to set Omaha equal to rows 2:53 of itself, e.g.

```
>>omaha=omaha(2:53,:);
```

and the other is to set row 1 equal to the empty vector, e.g.

```
>>omaha(1,:)=[];
```

I slightly prefer the former method as it seems less tricky.

14. Let's now create the vector of corn prices. As I mentioned in class, "(:)" will convert anything to a column vector. What I didn't mention is that this works for matrices as well. So, if arr2d is m-by-n, then arr2d(:) is m*n-by-1. How does this work? Well, Matlab creates the column vector by stacking the columns of the array. To see what happens, create a 2-by-3 sample array:

    ```
    >>arr2d=[1 2 3; 4 5 6]
    ```

    Then, type "arr2d(:)". What happens? For the omaha array, time increases down a column, so the (:) syntax should work. Assign the column-equivalent of omaha to a new array called omahav.

15. We now need to create a column vector the same size as omahav with time-labels for each sample. From before, we know that if we can create a matrix the same size as omaha with the time in each cell corresponding to the time the price was observed. In our case, time is a combination of years (the column labels) and weeks (the row labels). First, let's get the size of omaha:

    ```
    >> [m, n]=size(omaha)"
    ```

    The variables m and n will contain the number of rows and columns, respectively. Next, let's create a matrix YR with each cell containing the year the sample was taken. To do this, we'll use an outer product:

    ```
    >>YR=ones(m,1)*year;
    ```

    Look at a few entries in YR to make sure you understand its structure. Now, we need to work with weeks. First, create a column vector with the weeks corresponding to each row:

    ```
    >>weeks=(0:m-1)';
    ```

    We can then create a matrix using an outer product, of course:

    ```
    >>WK=weeks*ones(1,n);
    ```

    If we convert weeks into fractions of a year (by dividing by m) and add this to YR, we'll have the matrix we want:

```
>>YR=YR+WK/m;
```

Again, double check that you understand the structure of YR. Finally, we can create our vector times by flattening YR:

```
>>times=YR(:);
```

16. Whew! We now have the vectors we want, let's plot them. Matlab's basic line-plotting function is called "plot". Plot is very flexible and there are a lot of was to call it, ype "help plot" to learn more. For our data, we want to plot omahav as a function of time, so type

```
>>plot(time, omahav)
```

It's as simple as that.

Last but not least, assume you want to take a break for a while (I certainly do), but that you want to work with omahav and time at some later time. To save these vectors to a .mat file, type

```
>>save omaha omahav time
```

This will create a file called omaha.mat in your working directory. To prove to yourself that this worked, clear your workspace by typing "clear". Then, load the data by typing

```
>>load Omaha
```

If you type "whos" you should see both omahav and time.

17. To quit Matlab, type "exit."