Lecture 3 Outline: Plotting Data and Data Reading and Writing

* Style: Making your Scripts Readable
  + Programming languages are not made for computers to read--they are made for humans to read!
  + Readable Matlab code involves
    - putting comments above each line or group of lines that perform a task
    - Using vertical spacing to show which parts are conceptually together and apart.
    - Using double-comment %% section headers to break code into obvious pieces
* New Variable Type: Letters (a.k.a “Characters” or “Strings”)
  + Letters, called **characters**, and words, called **strings**, can also be stored inside Matlab variables.
    - To make a string, simply put the string you want to save inside **apostrophes (‘)**.
    - When the string is completed, it will turn purple in the Matlab editor and command window.
  + Indexing Strings: Indexing works exactly the same way in strings as it did in numbers. You can even have matrices of characters, and you index those the same way you did with matrices of numbers.
    - ex) myString = ‘The Dog Walked to the Park’; myString(5:7) --> ‘Dog’
    - Note that spaces are also considered to be characters
    - Logical Indexing works the same, too!
      * ex) myString(myString>’g‘)
      * Note that Capital letters are not equal to their lowercase pairs. ‘A’ is not equal to ‘a’.
    - Note: Because words are often different length and are not handled as a single “word” entity, vertical arrays of words (ie. character matrices) are difficult to handle. For now, avoid this by sticking to single letters when making vertical arrays.
  + Characters and numbers can not mix! A matrix must have only one type inside it.
    - ex) Try myArray = [3, ‘a’, 4 ,‘b’]. You get an error!
* Visualizing your data! Line graphs, Scatterplots, Histograms, and Bar Plots

Understanding the way Matlab thinks of Figures: breaking it down into:

* + - The Figure: this is the actual window that pops up when you plot something.
      * To make a new figure, use the **figure** function.
    - The Axis: this is the, well, axis that the data is plotted on, including the way that the axis is represented. This includes the axis tic marks and the axis labels.
      * To add a title, use the **title()** function.
      * To label your axes, use the **xlabel()** or **ylabel()** functions.
      * To change the range that is plotted (Matlab will do its best automatically, but sometimes it isn’t so nice), use the **xlim()** and **ylim()** functions.
    - The Plot: this is the data itself that is plotted, and how it is represented (the line/markers that are shown)
      * Depending on the type of plot you want to make, the function will differ. Following are high-level functions that will produce a figure, axis, and plot style in a single command:
      * **plot(myData)** produces a line plot, if there is only one input.
        + If the input is an array, it will make one line.
        + If the input is a matrix, it will make one line for each **column** in the matrix, automatically making each line a different color.
        + Class Exercise: plot a sine wave. Then plot two sine waves on the same plot (by first putting two arrays into a matrix, then plotting it).
      * **plot(myX,myY)** produces a scatterplot that is connected by lines.. Every point on the scatterplot is represented as a matching X,Y coordinate. There should be only one array in myX, but there can be as many columns of data in myY as you like. Matlab will automatically make each column in myY a different color.
        + Class Exercise: plot a sine wave that has useful x-data.
      * **plot(myX,myY,’o’)** produces a scatterplot that is **not** connected by lines. Everything else is the same.
        + Class exercise: plot random cloud of data using **normrnd()**. Show how messy it looks with and without lines, and the difference between a line graph and a scatterplot.
      * **hist(myData)** produces a frequency histogram.of an array of data as a bar graph.
        + Class exercise: make a histogram of the **normrnd()** data, showing that it is normally-distributed.
      * **bar(myData)** produces a bar graph, with a different bar for each number in an array, and clustering of bars for each row in a matrix.
        + Class exercise: make a bar graph

Drawing A Figure**:**

* + - In a script, Making a figure almost always follows the same pattern:
      * figure()
      * plot() (or whatever other of the plot functions you use)
      * title(), xlabel(), ylabel()
    - If you want to draw multiple plots on a figure, you need to have it **hold** **on** to the previous plot it drew, rather than overwriting it:
      * figure()
      * hold on
      * plot()
      * plot()
      * plot()
      * hold off
      * title(), xlabel, ylabel()

Making Your Figure More Attractive: Matlab’s default figures are, to put it nicely, fairly ugly--they need some modification to get to a state that you can publish them. Although Matlab comes with a wealth of functions and methods for doing so, we will focus only on using the **Figure Editor** and **Property Editor** in the Matlab IDE.

* + - Figure Editor: Allows direct access to the elements of a figure whose properties can be changed in the property editor.
    - Property Editor:
      * shows the basic properties of a figure elements
      * Note: can be slow if you are making a lot of figures, it does allow you to access all of the properties a figure can have and make it something you can be proud of.
    - Class Exercise (10 minutes): Take one of the figures made in the previous examples and customize it using the Property Editor. Edit the line style/colors, enlarge the font, add titles, etc.

Saving a Figure: To save a figure, click **File -> Save As** in the figure gui. If you want to save it as a Matlab figure that can still be edited in the Property Editor, save it as a .fig. If you want to save it as an image for use in presentations or papers, select a different image format in the dropdown menu.

* + - Note: different image formats have different advantages and disadvantages for different types of figures. We won’t be discussing them here, but please experiment with different ones to ensure you get something that looks pleasant when used in your presentation.
* Reading and Writing Data:

It’s great that we can make data inside Matlab, but in our research, what we really want to do is take data that we obtained somewhere else and analyze it inside Matlab. Often, we further want to save our analyzed data to be used in later Matlab analysis sessions!

There are as many different reading and writing functions as there are filetypes. Often, you’ll have to ask someone in your lab what the correct one is for your data. Here, we will discuss two common types of formats that you can use as general-purpose solutions for most basic data formats: Matlab data files and comma-deliminated text files.

Matlab GUI: The “Import Data” Button:

* + - The Import Data button opens an interactive wizard that will help you import most types of data.

Matlab’s .mat Files:

* + - Read: **load(‘filename.mat’)**
    - Write: **save(‘filename.mat’,’varName1’,’varName2’, etc)**
    - Matlab’s .mat files are only useful in Matlab! Still, they are a convenient way to save your variables for later Matlab sessions.
    - Warning: If no variable names are provided after the file name, then the function assumes you want to save your entire workspace! This can be good (maybe you want to save your whole session for the next day), or bad (do you really want to save 30 temporary variables?)
    - Class Exercise: Save the **normrnd()** data from earlier into a .mat file. Exit Matlab, then load the data again in a new session. Yay! No data was lost!

Something-Deliminated Files:

* + - Read: **data = dlmread(‘filename’,delimiter,RowNum,ColumnNum)**
    - Write: **dlmwrite(‘filename’,myData [,’-append’])**
    - A delimited file is simply a text file in which the numbered data points or string labels are separated into different rows and columns by some delimiter. Often, this is a comma, tab, or space. A file with a comma as a delimiter is called a comma-delimited file.
    - The file’s extension doesn’t matter to Matlab here, although Windows will prefer it to have the .txt extension for opening in a text editor.
    - Demo: Make a comma-delimited file in a text editor and save it for later use.
    - Note: When reading the data, you must remember that strings and numbers need to be separate variables in a matrix. If your data file has both, you need to load them seperately, specifying which rows and columns to use (or, at least, which to start from).
      * The same goes with writing. To make headers, just write your string matrix, then write your data matrix and include the **‘-append’** input to not overwrite the headers.

Homework: Loading Comma-Delimited Data and Plotting It

This assignment requires the data **that has been passed around on a USB stick**. Be sure to copy it onto your hard drive before you leave the lecture!

There are two files that you’ve been given: **matlab\_lec3data.dat** and **matlab\_lec3data.mat**. These are both the same data, but in different forms. In this assignment, you should use the .dat file, but if you get stuck, you may use the easier-to-use .mat file instead so that you can complete the analysis and plotting portion of the assigment.

Description of the Experiment: 8000 trials of short auditory tones were played to a participant, and consisted of either a loud tone (condition 1), or a quiet tone (condition 0). One electrode of EEG data was recorded 40 msecs before the tone, and 65 msecs after it. We are interested in how the average EEG waveform changes over time in response to the auditory tone.

Description of the Data: The .dat file contains a single matrix that contains:

* the time points (the first column, from the 2nd to last row),
* the condition number of each trial (the first row, from the 2nd to last column)
* the EEG data: each trial is its own column, from the 2nd row to the last row.

Assignment:

* Load the data and seperate it into time, condition, and data variables.
* Plot the mean of each condition’s EEG waveform data over time..
* Edit the plot to make it more interesting to look at.

