PHY 477/577: Obs. Methods & Data Analysis in Phys. Ocn.
Assignment 1: Reading & Manipulating Data

Derek J. Grimes
grimesdj@uncw.edu

Goal: develop familiarity with managing, reading, interrogating, and manipulating coastal data.

Expectations I anticipate a wide range of coding/programming backgrounds. If you've never used matlab, this is your opportunity to learn or improve. If you're a matlab guru, try python or R. Or consider drafting assignments using LaTeX, making publication quality plots, or learn version control using GIT.

Task: Load the SIO pier temperature & salinity records downloaded from: https://library.ucsd.edu/dc/object/bb4003017c. Generate time series plots and estimate basic statistics for surface & bottom temperature from the >100-year-long record. You should save your code to a file 'LastName_FirstName_HW#.m' and your responses to the below questions in a matching .pdf filename. You can alternatively combine your responses and code in a single .pdf using latex or similar.

1. Download the La Jolla temperature/salinity ASCII files (.csv). You can either do this manually, or develop code to download data directly from the url. For example, use the sample code below as a starting point (note: the % symbol tells matlab that text on the remainder of the line is a comment not code).

```
% download 100-years of observations at the SIO pier from url:
 url = 'https://library.ucsd.edu/dc/object/bb4003017c/_1_1.zip/download';
%
% unzip the contents into your data directory
 unzip(url,'/path/to/data/');
```

2. Open the file using a text editor, like TextEdit or Notepad. Is there a header describing the contents? What information (meta-data) is important to interpreting the contents?

3. Read the file contents into matlab. The necessary steps are:

   (a) generate a pointer to the file on your system using *fopen*(),

   (b) count the number of header lines above the data stream,

   (c) determine the character separating columns, called the delimiter (*e.g.,* {' '} for space, {','} for comma, etc.),

   (d) the column data format (*e.g.,* '%f' for float, '%u' or '%d' for unsigned/signed-integer, '%s' for string/characters, '%*s' means ignore character column),

   (e) determine the character used to fill empty/erroneous values,

   (f) Read the file contents into a cell-array by passing these to function *textscan*().

   Here is some example code... you fill in the '?????'.

```
 dataPath = dir(['/path/to/data/LaJolla_TEMP_*.csv']);
 filePath = [dataPath(1).folder,filesep,dataPath(1).name];
% you should first open the .csv file to get number of header lines
% and format of the data stream.
 nHeaderLines = ????;
%
% To use matlab's text-scan function to read in the remaining data fields,
% first we need to tell it the data-type/format of each column,
 format     = '%u %u %u %f  ????';
```

3. b). rows 0-45 make up the header lines above the data stream.
c). Assuming comma as the delimiter as it is a csv file. However, if the date is separated by commas then the delimiter must be different.
e). Nan is sued to fill empty/erroneous values
f). not sure how to read file contents into a cell-array.

2.Yes, important data includes where the data was collected, when the data was collected, who collected the data. Additional information about the data was provided: NaN (data not collected), 0(good data), 1(illegible entry), 2(data differs from other sources, 3(data uncertain) 4(leaky bottle), 5 (sample collected at different location). Its important to note the default for the header code, df.header is the first five rows but can be changed easily for example to 10 with, df.header(10). #df=data_frame

```
% and the delimeter separating each column,
 delimiter = '????';
% and what to skip over as missing data
emptyVal   = '????';
%
% now generate a pointer to the start of the file on your file system
 fid        = fopen(filePath);% fid points to the first bit of the file
 data       = textscan(fid,????,'delimiter',????,'treatasempty',????);
```

4. Extract data from the cell-array, allocating them to appropriate variable names. Use a logical shorthand, such as, *tempSurf* for surface temperature and *tempBott* for bottom temperature.

5. What are the relevant dependent and independent variables and what are their units, data class (single, double, unsigned integer), and resolution (*i.e.,* Δt)?

Dependent Variable depending on the data set is surface/bottom temperature, or surface/bottom salinity. the independent variable for both data sets is the date (time).

6. How many observations are there (*nObs=????*)?

7. Is there a quality control flag? Are there any *NaN*s, which stands for not-a-number? Use function *isnan*() to locate and mask the bad data. Also, determine the number of *NaN*'s (*nNaN = ????*). What fraction of the observations are valid?

$$\Delta_1 = \frac{\mathrm{nObs} - \mathrm{nNaN}}{\mathrm{nObs}},$$

Not sure how to do as I cant even create an array.

8. Manually compute the means ($\bar{T}_\mathrm{s}$, & $\bar{T}_\mathrm{b}$) and standard deviations (s_s & s_b),

$$\bar{T} = \frac{1}{N}\sum_{n=1}^{N} T_n,$$

$$s = \frac{1}{N-1}\sum_{n=1}^{N}(T_n - \bar{T})^2,$$

To find the mean; add all of a given variable together and then divide by the number of variables. To find the SD find the mean, then for each Indvidual variable value find its distance from the mean, by subtracting it from the mean. The sum all of the previous values together and divide it by the total number of data points. Lastly take the square root.

Use basic commands only,*e.g.,* $\pm$, *sum*(), ().^2 (the '.' indicates element-wise operation), and either ()^0.5 or *sqrt*(). What normalization factor N did you use? Hint: something to do with *NaN*s

9. Save your entire matlab workspace as a binary '.mat' file. What is the fraction difference in file-size between the original ASCII file and the binary file? Use the equation,

$$\Delta_2 = \frac{\mathrm{ASCII} - \mathrm{MAT}}{\mathrm{ASCII}},$$

where ASCII and MAT are the respective file-sizes measured in bytes.

10. Plot the time series for temperature with surface colored red and bottom colored blue. Use command *datetick*() to label the abscissa ticks based on 'yyyy'. Label all of your axes with appropriate units using *label*(). It should look something like Figure 1 in https://doi.org/10.1029/2019JC015673. You can refer to the code in Lecture 2 notes.

11. Describe any notable features/patterns in the graphical data (2-3 sentences).