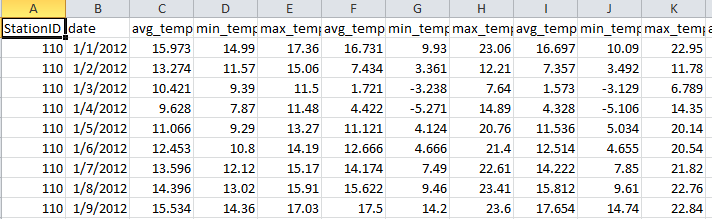
**Weather Data Project**

FAWN “Florida Automated Weather Network” is a series of 36 weather stations spread throughout the state of Florida that collect various types of weather data. A sample zip file containing comma separated value (csv) files, with FAWN data is provided. The csv contains weather data attributes such as temperature, rainfall, and so forth. In its current state, the csv file can’t be automatically uploaded into ArcGIS and visualized as weather data. The csv file has to be modified and then joined with a spatial dataset. Only after this can geoprocessing be performed. This project involves automating this process of downloading data, unzipping it, preprocessing the csv files, joining it with spatial data, and finally, performing geospatial analysis on the data.

This project can use the FAWN data site (search for ‘FAWN Florida weather’ and look for the yearly csv data under ‘data access’) **OR** another found data clearinghouse that poses parallel challenges. The application must download the data and unzip the data, then modify the csv file as described below:

* Remove lines that only contain carriage returns (‘\n’).
* Remove unneeded columns (needed ones will depend on user input).
* Remove rows in which a value is missing (The sensor might have mal functioned that day).
* Select rows with dates between user input start and end dates.
* For each station id, find the average value over the given time interval for the user input attribute of interest (for example AvgAirTemp, Rain, ET, etc.). As an extra challenge, the program could automatically populate the list of choices in the GUI based on the input fields.

If the dates span more than one file, the resulting csv files should be combined. The overall modified csv output file should be saved. Next, some conversion calculations should be made for the attribute of interest given by the user. If the attribute of interest is a measure of temperature, a new field should be calculated containing the Celsius to Fahrenheit conversion (or vice versa) and added to the shapefile. If the attribute of interest is a measure of distance, a new field should be calculated for the metric to English unit conversion (or vice versa e.g., mm to inches) and this should be added to the shapefile. The csv file then needs to be joined with a shapefile (such as floridaStations.shp) so the data is attached spatially to the location that it represents. Next, the goal is to have the data interpolated and displayed over the state of Florida (or other area if you use a different data source). Perform a mask so only the area of interest is shown (e.g., Florida) polygon or polyline shapefiles can be found (for free) on-line. Finally, the shapefile should be added to a map and an output HTML report should be generated. In summary, the following outputs should be created:

* A cleaned csv file only containing needed data, saved in the output directory.
* A modified shapefile with the station averages for the attribute of interest.
* An HTML report for the input file. The HTML report should at a minimum, show the name of the attribute of interest, the date range, and an image of the resulting map (the arcpy.mapping module allows you to export an image). As an additional challenge, you can also try to add a table containing one row for each station, giving the name and attribute average value.

The resulting csv, shapefile, and HTML files need to be saved in a new folder, designated by the end user. Within ArcMap a button needs to be created. This button will launch a GUI interface (created with a script tool), prompting the user for several pieces of input information. Allow the user to select:

* A site where a csv zipfile can be downloaded.
* The output location of the new folder where the output will be saved.
* Start and end dates (use the datetime module to process these).
* An attribute of interest (e.g., AvgAirTemp, Rain, Evapotranspiration (ET), etc.)
* Fahrenheit or Celsius, if the selected attribute is a measure of temperature, or if the attribute is a measure of distance, allow the user to select mm or inches

The best way to approach this is to open the csv file, read the contents, modify the contents and create an edited csv that can be joined with a copy of the shapefile. Then the geoprocessing and calculations can be executed. Use the file open/read/close functions you learn in class, not the csv module.

Complete the csv processing and shapefile join generation for the ‘project core Python code’ submission. You don’t have to have a button and GUI yet at that point, but your code should be handling the user input via the system arguments by then.

As you start the project, you will not yet know how to automatically add data to the map, how to write an HTML file, or how to make a button and a graphical user interface to get input from the user. But you will already know how to do the geoprocessing involved in this project. Start by creating a smaller csv that you ‘fix’ by hand. For your test file, keep only a few rows, enough so that you can calculate some averages. Get your code to do the spline and masking. Next, get your code to compute the conversion fields for some dummy values. Next, you’ll learn how to read and write text files in class. The csv is a text file (with commas separating the column values). So the next step is to redo the csv file fixing that you did by hand with your script and then compute the averages, so that they can be joined with the shapefile. Next, make your script flexible by using arguments instead of hard-coding the input variables, such as the attribute of interest and so forth. Finally, you’ll need to create the button, GUI, and output HTML.

\*Project was developed by Brian McLean.\*