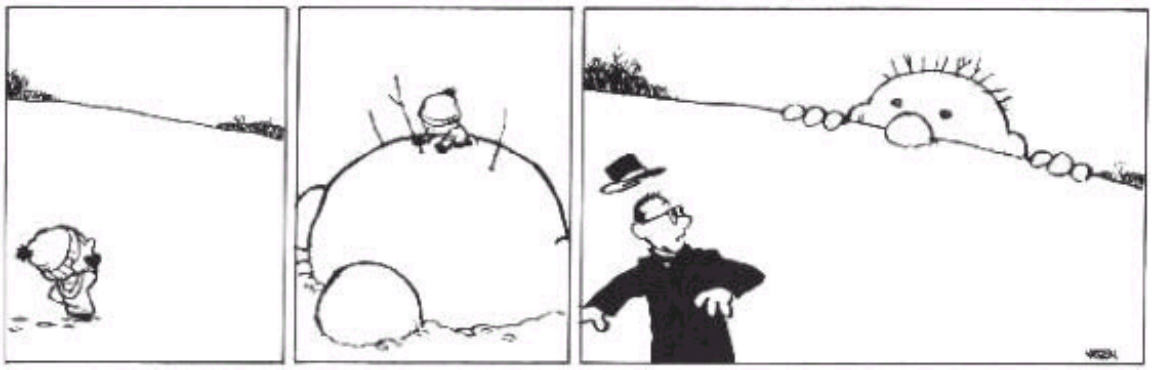
CS150 - Assignment 11
Weather Revisited
Due: Wednesday Dec. 4, at the beginning of class



http://www.stargazersrealm.com/MAIN/funnies/calvinhobbes.html

For our final lab, we're going to be adding some additional functionality to our weather program from Lab 9. The goal of Lab 9 was to write a program that could collect weather data over time. Once you have this data, the next step is to analyze it! This lab explores this data analysis using Matlab.

1 Function recap

Just to make sure everyone is clear, here is a quick review of defining functions in Matlab:

- To define a function use the following syntax:

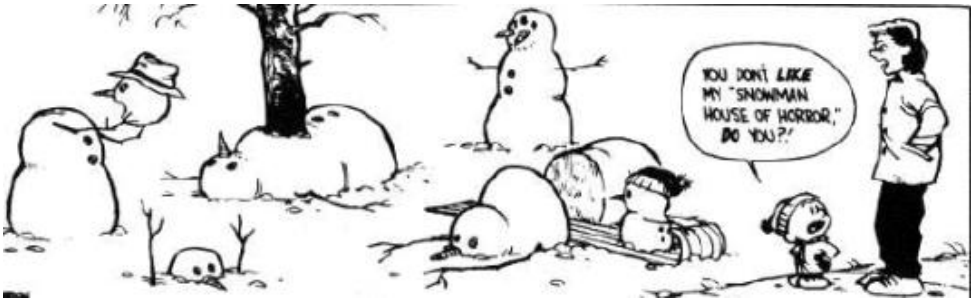
  ```
  function return_var = function_name(parameters)
  ```

  - `function` is a keyword (like `def`), indicating that we are defining a function
  - `return_var =` indicates that we will be returning a value from this function. When the function finishes executing, whatever value is in this variable (which you can name whatever you want) will be returned by the function. If you do not need to return any value, you should leave this portion off.

- You must put the function in a file will the same name as the function with a .m after it. For example, if you're writing a function called `plot_weather`, then it needs to go in a file called `plot_weather.m`.

- This means that generally speaking we will define one function per file (you can define others, however, they will not be visible/callable outside of that file).
- You *should* include a "docstring" for each function which is a block of comments immediately following the function header.
- The `function...` line should be the first line in the file. Do *NOT* put any comments/docstring above the function (only for scripts) since this will mess up the `help` functionality for the function.
- You do **not** run a function file like you do in Wing (i.e. by pressing the run button). As long as the .m file is in your current working directory, you can just call the function normally.

2 Weather data



Reading and writing from files in Matlab can be done easily if the data is represented in the file in a matrix-like format. To get you started on this assignment, I have posted some weather data in a format that can be read by `dlmread` in Matlab. At

http://www.cs.middlebury.edu/~dkauchak/classes/cs150/assignments/assign11/

there are two files that consist of space delimited data:

- `test.matrix.txt`: A short file with just a few days of data that will be useful for checking your answers on. This data was synthesized from `test.raw.txt`.
- `wisc.matrix.txt`: Real weather data aggregated for roughly 1 year from Wisconsin Rapids, WI obtained from:

  http://www.soils.wisc.edu/asigServlets/asos/SelectHourlyAsos.jsp

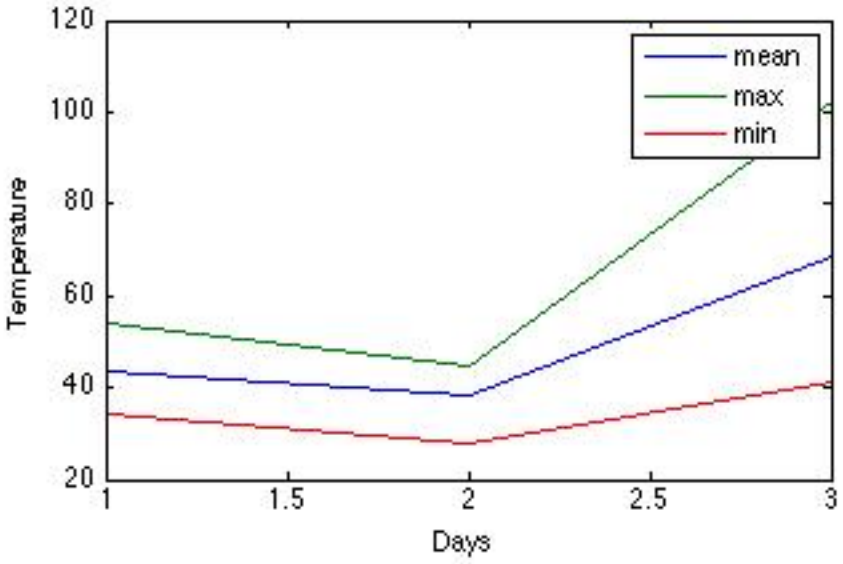
  This data was synthesized from `wisc.raw.txt`.

Each line in these files has 24 temperature readings separated by a space representing hourly temperature readings for one day. Download both of these files into your working folder for this lab. Take a look at the data in these files and make sure you understand what the data represents.

3 Visualizing the data

In Matlab, write a function called `plot_weather` that takes a matrix of weather data as a parameter and then plots the average, maximum and minimum temperature for each day in the data. Your plot should include appropriate labels for the x and y axis and should also include a legend. The input matrix will be in the same format as the data file, with each row having 24 entries and each row representing a day.

For example, a plot of the test data should look something like:



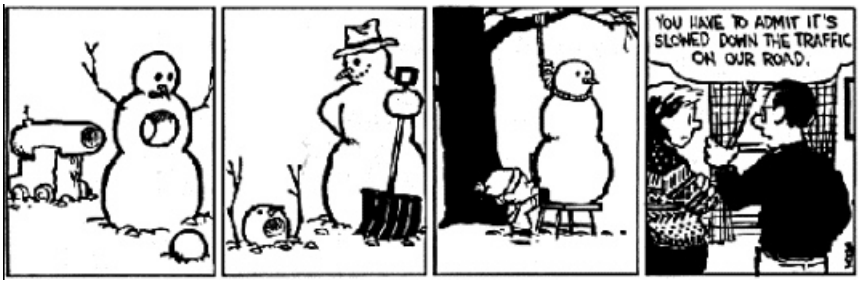
A few hints:

- When testing your function, you can read in the data from one of the `.matrix` files and then pass it to your function.
- See `help legend` for more information about adding a legend.
- Most of the built-in functions in Matlab summarize data along the columns, so it may be helpful to transpose the data matrix (denoted with a single quote).

When you've got it working, try plotting the weather data from Wisconsin. The data should look much more interesting since it's real data! Can you figure out around what time the data set starts and ends?

Save a copy of the plot of the Wisconsin data and submit it along with your code. To save the plot, click on the "Save" button (it looks like a disk) in the plot window. Under file type, select ".jpg".

4 Analyzing the data



Plotting the data allows us to analyze some aspects of the data, but there are many interesting additional questions that we could also ask. Write a function called `weather_stats` that takes a matrix of weather data as a parameter and displays the following information about the weather data:

- The number of days in the data set.
- The overall average temperature.
- The coldest temperature in the data set.
- The hottest temperature in the data set.
- The average daily temperature fluctuation. The daily temperature fluctuation is the difference between the hottest temperature and the coldest temperature. The average daily temperature fluctuation is the average of this fluctuation over the days.
- Number of days where the temperature was above 100 degrees.
- Number of days where the temperature was below 32 degrees.
- Number of days where it didn't get above freezing (i.e. 32 degrees).
- Average temperature on those days when the temperature got below 32 degrees.
- One additional interesting fact about the data of your choosing.

In writing this function:

- For at least one of these calculations you must write another external function (i.e. in its own file) that returns a value and that you use within your `weather_stats` function.
- You should also print labels for each data item so it's clear what the data represents.
- You should try very hard to avoid `for` loops. Most of these statistics can be written straightforwardly using matrix operations with just one or two statements in Matlab (in fact, all of them can be written without `for` loops, though one or two are trickier). You will not receive full credit if you use `for` loops extensively.

For example, the output from this function on the data from the test file would be:

```
Days in data set:
     3

Average temperature:
   50.0833

Coldest temperature:
    28

Hottest temperature:
   102

Average temperature fluctuation:
   32.6667

Number of days above 100:
     1

Number of days below 32
     1

Number of days were it didn't get above freezing:
     0

Average temperature when temps got below 32:
   38.5417
```

A few hints:

- As with the previous function, it may be helpful to transpose the data matrix (denoted with a single quote).
- Test each of these statistics one at a time.

- If you get stuck, try and work out a small example by hand.
- Sometimes it can be helpful to write the statistic using a `for` loop and then once you have it working, try and figure out a way to write it without the loop.

5 Obtaining the data

To utilize our programs in a real scenario, we still need one important program: a program that converts the raw temperature files that are output from our Lab 9 program into the delimited format that Matlab can handle. You could try and do all of this in Matlab, however, it is not well suited for it.[1]

Write a Python program called `convert_raw_weather.py` that takes two *command-line arguments*. The first argument should be the name of a raw weather file to read data from and the second argument should be the name of a file to output the delimited weather data to. If imported, your program/module should not do anything. For example, to generate `test.matrix.txt` from the raw data in `test.raw.txt` you would run the following in `Terminal`:

```
python convert_raw_weather.py test.raw.txt test.matrix.txt
```

This program should be an extension to your lab prep work.

6 Extra credit

You may earn up to 2 points of extra credit on this assignment. Below are some ideas, but you may incorporate your own if you'd like. Make sure to document your extra credit additions in comments at the top of the file.

- Write an additional statistic regarding the weather data.
- Plot additional weather statistics over the days. You can either plot them on the same plot, or write another function to plot different statistics.

[1] You can nail in a nail with a wrench, but a hammer works better...

- Write all of your Matlab code without using *any* for loops.
- Add your own...

7 When you're done

Make sure that your program is properly commented.

For Python

- You should have a docstring comment for each module at the top of the file.
- You should have comments after the module docstring stating your name, course (including section number), assignment number and the date.
- Each function should have an appropriate *docstring*.

For Matlab

- You should have a docstring comment for each function.
- You should have comments after the function docstring stating your name, course (including section number), assignment number and the date.

In addition, make sure that you've used good ***style***. For Matlab this includes avoiding using `for` loops.

Submission procedure

What to submit

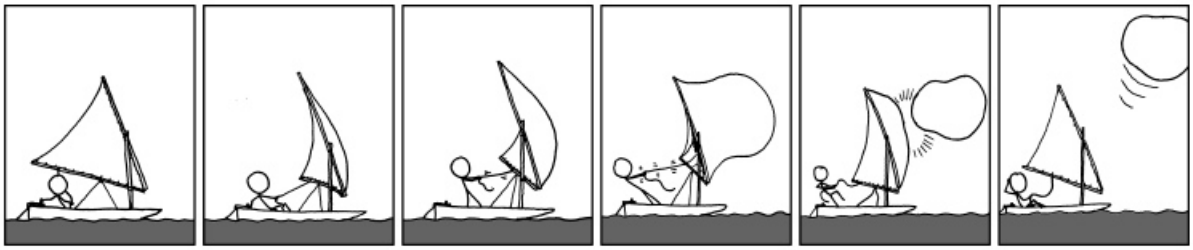
- All of your .m files (you should have at least 3)
- The .jpg plot file saved from plotting the Wisconsin data
- `convert_raw_weather.py`

Create a directory with your name followed by the lab number. For example, my folder would be called `davidkauchak11`. Put all the files inside this directory and the create a .zip file from this directory. On the Macs, right-click on the directory and select "Compress...". If you're working on Windows, right-click on the file and select "Send to" then select "Compressed (zipped) Folder" (or if you don't have this option, use http://www.winzip.com/). You will then see a file with a .zip extension be created.

Submit this .zip file digitally at the usual location:

http://www.cs.middlebury.edu/~dkauchak/classes/cs150/submission/

Enter the relevant information and upload your file. If you have problems with this, please let me know. This link can also be found in the "Resources" section of the course web page at the bottom.



http://www.xkcd.com/976/

Grading

		points
`plot_weather`		
	generates appropriate plot	4
	legend and labels	1
`weather_stats`		
	days	0.5
	ave. temp	0.5
	coldest temp	0.5
	hottest temp	0.5
	ave. temp fluctuation	1.5
	days above 100	1.5
	days below 32	1.5
	days not above freezing	1.5
	average temp when below 32	2
`convert_raw_weather.py`		
	works properly	4
	command-line	1
Comments, style		5
extra credit		2
total		25 (+2)