HW2: Analysis of Temperature Data

CSE1010 Fall 2012
Jeffrey A. Meunier
University of Connecticut

1. Introduction

In this assignment you will write a Matlab program to perform some analytical calculations on a vector of numbers. The numbers are a collection of temperature readings from an airport in Birmingham AL in 2003. You will create a graph of the data in order to visualize the data.

2. Value

This program is worth a maximum of 20 points. See the grading rubric at the end for a breakdown of the values of different parts of this project.

3. Due date

This project is due by 11:59PM on Sunday, September 16, 2012.

4. Objectives

The purpose of this assignment is to have you improve your skills in these areas:

- Creating and running a Matlab program file.
- Using vectors.
- Using simple built-in Matlab functions.
- Visualizing data on a graph.

5. Background

The information you need for this and future projects is mostly in chapter 3 in the book. The data analysis functions that you will need are in section 3.4. However, the **find()** function is not located in chapter 3. You will need to locate this information on your own and do your best to figure out how to use it. Real world problems never follow the sequence of topics in a book.

6. Assignment

Read through all of this document before you start to make sure you understand how you must proceed with the assignment. For this assignment you will create a program that generates output in the console window and also displays a graph. Then you will collect your program, data, and graph into a brief report.

6.1. Create a new project folder

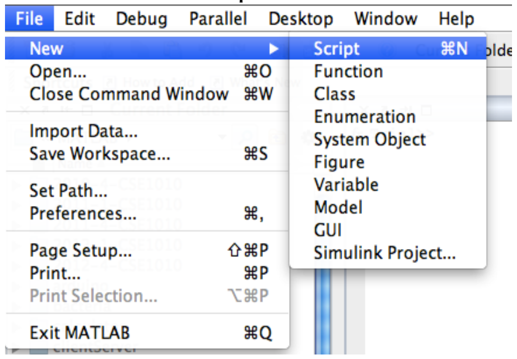
In your CSE1010 folder in Matlab create a new folder called HW02 or Project 2 or Assignment 2: something to distinguish it from all the other assignments you will do this semester.

If you do not have a CSE1010 folder in Matlab, now is the time to make one. DO NOT place all your programs in one single folder.

To make a new folder in Matlab, right click in the Current Folder window and select New folder.

6.2. Create allTemps function

Create a new script file.



Even though you're creating a new function, choose *File* → *New* → *Script*. In the edit window that pops up, type the function shown below and save it as **allTemps.m** in the current folder (be sure to capitalize it like that: small **a**, big **T**). This is the function:

```
function T = allTemps
  T = [59 57 57 57 57 57 56 55 55 55 55 54 54 54 54 54 54 54 54 52 50 50 50 ...
       50 50 50 49 48 49 48 48 48 47 47 47 46 46 46 46 46 46 46 46 46 46 46 ...
       46 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 ...
       45 46 46 47 48 48 48 49 49 49 48 50 48 49 48 46 45 45 43 43 42 42 39 ...
       38 37 36 35 35 34 34 34 34 34 34 34 34 34 34 33 34 34 34 34 35 34 34 ...
       33 33 34 32 31 29 29 28 28 26 27 26 25 25 28 31 35 38 40 44 45 48 48 ...
       46 44 44 43 42 43 43 43 44 45 46 46 44 42 40 45 46 49 52 55 58 60 60 ...
       59 55 54 51 46 43 41 40 39 37 36 36 36 34 32 32 35 40 43 43 43 43 43 ...
       42 42 42 41 38 38 35 34 34 33 31 31 29 30 28 25 27 23 28 31 34 38 41 ...
       45 47 47 47 45 42 41 40 38 37 37 36 36 35 34 35 35 33 35 37 40 44 48 ...
       52 57 60 60 60 57 56 54 52 53 52 51 49 48 49 49 48 48 48 48 49 53 57 ...
       59 62 66 68 67 63 62 62 59 58 55 55 55 55 54 54 54 54 53 54 52 51 50 ...
       50 48 45 43 38 37 38 40 42 43 45 47 48 48 47 45 43 41 38 37 34 32 31 ...
       30 28 28 27 27 27 25 28 29 31 35 37 39 40 39 38 37 35 35 35 34 34 32 ...
       31 29 30 30 28 27 27 26 27 28 29 30 32 33 34 34 34 33 32 32 31 28 29 ...
       29 29 29 29 30 30 30 30 30 31 32 34 37 38 42 45 47 47 43 40 35 33 33 ...
       30 30 29 29 27 29 27 27 27 27 31 37 42 46 49 53 54 54 54 51 48 41 38 ...
       37 37 36 34 33 32 29 28 26 26 26 24 28 31 35 38 40 43 44 45 45 40 40 ...
       38 31 33 32 34 34 34 35 31 32 29 34 37 38 40 41 40 39 40 39 41 41 41 ...
       41 NaN 41 NaN 43 44 45 45 45 44 45 43 40 37 37 37 35 34 32 31 30 29 29 ...
       29 28 28 27 27 27 28 28 28 28 28 28 30 31 32 34 35 36 35 33 31 30 27 ...
       23 23 22 20 19 18 17 17 17 14 15 18 21 25 27 30 33 35 36 36 37 35 36 ...
       36 36 35 31 29 27 26 25 27 25 25 22 28 30 34 37 39 43 45 47 46 45 41 ...
       41 41 41 43 44 44 44 42 42 41 40 41 41 43 42 43 45 48 53 56 57 57 58 ...
       59 60 60 58 58 58 58 58 58 58 57 57 57 57 56 55 55 55 56 55 55 56 57 ...
       58 60 61 61 63 63 64 64 63 55 54 52 48 47 46 46 45 45 45 45 45 43 43 ...
       43 43 43 43 42 42 41 39 40 39 39 37 38 37 37 36 36 36 35 35 36 40 43 ...
       46 49 50 51 50 45 41 38 36 34 32 31 29 28 26 25 23 21 21 19 18 16 17 ...
       18 17 19 21 23 23 24 23 23 20 19 18 17 16 15 14 12 11 10 9 8 7 7 7 9 ...
       13 18 21 24 26 29 32 32 30 27 25 24 25 24 24 23 24 25 24 21 22 23 23 ...
       23 27 30 33 34 36 37 38 37 37 36 36 36 36 35 35 35 34 34 34 34 34 34 ...
       34 34 35 37 40 44 46 48 49 48 47 44 43 42 39 37 32 30 28 28 26 25 22 ...
       21 20 19 18 21 26 29 33 36 39 41 43 43 43 41 38 29 32 30 30 33 33 30 ...
       28 31 34 36 38 40 41 44 47 50 53 55 56 56 55 55 54 55 53 55 56 55 53 ...
       53 53 53 53 53 53 53 54 55 55 55 55 56 56 57 56 55 55 55 55 55 55 55 ...
       55 55 55 56 55 56 56 55 55 56 56 55 NaN 55 NaN 55 NaN 52 51 48 48 46 46 ...
       47 45 45 45 44 45 43 43 42 41 39 40 39 40 39 40 40 41 41 42 43 42 43 ...
       43 40 39 39 39 39 39 38 38 38 37 37 39 39 38 38 38 38 38 39 40 39 43 ...
       43 47 50 53 55 55 52 49];
end
```

jk lol. I hope you didn't actually type all that. Copy and paste the function shown above into that file, then save the file and close it. Don't worry about how functions work just yet. We'll get to that in a few weeks.

Another way to do it is to type **edit allTemps.m** in the command window, paste the function, save the file.

Close the edit window when you're done with it.

If copy & paste isn't working for you, go to the Google web site for this course and find the file to download for this assignment. Send me e-mail if you don't see it.

6.3. Create a main.m script file

Create a new file in your project folder. It should be a script file called **main.m**. Place a comment block in the file that contains this information:

```
% Analysis of Temperature Readings
% Project 2, CSE1010 Fall 2012
% today's date (replace this line with the date)
% your name (replace this line with your name)
% your section number (replace this line with your section number)
% the name of your TA (replace this line with the name of your TA)
% Instructor: Jeffrey A. Meunier
clc     % clear command window
clear   % clear all variables
```

The rest of the statements for this program will be typed in this file below the *clear all variables* line.

6.4. Store temperatures in a variable

When you call the **allTemps** function, it returns a vector that you can store into a variable (type **allTemps** at the command prompt to see what I mean). Type this line into your Matlab program:

```
     Temps = allTemps;
```

Now when you need to work with the vector of temperatures, you will use the vector stored in the **Temps** variable rather than calling the **allTemps** function.

Now write two statements in the **main.m** file to tell the user how many temperature readings there are:

1. Use the **disp()** function to display a message. In my program I wrote this:
```
 disp('Number of readings')
```
2. Display the length of the **Temps** vector. If you call the **length()** function on **Temps** and leave off the semicolon, it will do what you need. Type those statements now at the command prompt to see how they works, then enter them into your program file.

When I run the My output looks like this:

```
  Number of readings
  ans =
     884
```

Actually there are blank lines between those lines of output, but I don't show them here because it takes up too much space. It really looks like this:

```
  Number of readings

  ans =

     884
```

6.5. Repair the temperatures vector

The vector contains the value **NaN** in some locations. In Matlab this means *not a number*. This could happen if the temperature reading was not available at that time, perhaps because the thermometer was malfunctioning. Scientifically speaking, this is the correct way to proceed: you must not use a number if no number is available.

However, Matlab refuses to do certain calculations on a vector if it contains **NaN** in any location, so we will replace each occurrence of **NaN** with a most likely expected value, which is just the average of the two temperature readings on either side of the **NaN**. For example, if a vector contains **[40 NaN 50]**, you would replace the **NaN** with 45, making the vector **[40 45 50]**.

Try to follow along with what I explain here. At the end of the section I'll tell you what to type into your Matlab program.

Type this into Matlab:
```
  Nans = find(isnan(Temps))
```
This tells you all the vector locations that contain **NaN** instead of an actual number. It also stores those locations into the vector stored in the **Nans** variable. It should look like this:
```
  EDU>> Nans
  Nans =
    439   441   820   822   824
```
This means that the **NaN** entries are at index locations 439, 441, 820, 822, and 824. We'll just have to trust it, because there are just too many numbers to count.

Now the vector elements in those locations must be changed to actual numbers. It's simplest just to replace each **NaN** with the average of the two numbers to either side of the **NaN** value.

For example, the first two NaNs occur in this series:
```
  .. 41 NaN 41 NaN 43 ...
```
Thus, the first **NaN** should be replaced by (41 + 41)/2 = 41, and the second should be replaced by (41 + 43)/2 = 42.

There are several **NaN**s in the temperatures vector, and it would be nice if you didn't have to locate and correct each one manually. It is possible to assign values to all those locations at once using a single assignment statement like this:
```
  Temps(Nans) = ...some calculation...
```
but what's on the right hand side of that assignment statement must have exactly the right number of values (it must have the same number of values as **Nans** has). You must create a vector of averages instead of just a single average. Each of those averages will be stored in **Temps** at its respective location.

Another useful function is the **mean()** function. It will calculate the mean (or average) of several numbers, or calculate a vector of means of two vectors of numbers arranged into two rows of a vector:
```
  EDU>> mean([1 2 3])
  ans =
      2
  EDU>> mean([1 2 3 ; 5 6 7])
  ans =
       3     4     5
```
This means that the mean of 1 & 5 is 3, 2 & 6 is 4, and 3 & 7 is 5.

Now observe:
```
  EDU>> Temps(Nans)
  ans =
     NaN   NaN   NaN   NaN   NaN
```
Recall: **Nans** contains the index locations where **Temps** is equal to **NaN**. Thus, **Temps(Nans)** gives you the values in the **Temps** vector at all those locations.

Now observe:
```
  EDU>> Nans-1
  ans =
     438   440   819   821   823
  EDU>> Temps(Nans-1)
  ans =
    41    41    55    55    55
```
Nans - 1 is an expression that subtracts 1 from each element in **Nans**, which then is a vector of all the locations *before* (or to the left of) the **NaN** entries, and those temperatures are 41, 41, 55, 55 and 55. You can do the same with **Nans + 1**. Those are all the locations *after* (or to the right of) the **NaN** entries. Now if you just average the values of the *befores* & *afters*, you have the new values that need to be stored in place of the **NaN** values.

```
EDU>> Means = mean([Temps(Nans-1); Temps(Nans+1)])
Means =
   41.0000   42.0000   55.0000   55.0000   53.5000
EDU>> Temps(Nans) = Means;
EDU>> Temps(Nans)
ans =
   41.0000   42.0000   55.0000   55.0000   53.5000
```
Now all the **NaN** values have been replaced by actual temperatures!

I will summarize what you must type into the **main.m** script file. Type statements to do this:

- Store the result of calling **allTemps** into the **Temps** variable. Use this statement:
  ```
  Temps = allTemps;
  ```
- Calculate the **Nans** vector (it's just what you did in the explanation above).
- Calculate the **Means** vector.
- Store the **Means** vector back into the **Temps** vector at the locations in **Nans**. This statement appears above. There are actually a few statements above ;) so you need to know what I'm talking about here in order to choose the right statement.
- Go back and place semicolons after all the assignment statements.

6.6 Calculate max, min, mean

Add these statements to your **main.m** script file, just below the statements you wrote previously.

- Show the maximum value of the **Temps** vector. Use the **max()** function. If you omit the semicolon it will display the value for you automatically. This is sufficient for this project.
- Show the minimum value of the **Temps** vector. Use the **min()** function.
- Show the mean value of the **Temps** vector. Use **mean()**.

Before each of those values, use the **disp()** function to display a message indicating whether the value is the maximum, minimum, or mean. Mine looks like this:

```
Maximum temperature:
ans =
    68
Minimum temperature:
ans =
    7
Mean temperature:
ans =
    39.9338
```

6.7. Locate temperatures below freezing

Create a variable called **Freezing** that contains the locations where the **Temps** vector is less than freezing (use 32 as the freezing temperature since the vector temperatures are in Fahrenheit). Use a semicolon because you do not need to show the values in this vector.

Use the **find()** function to find these values.

Use the **length()** function to calculate and display the number of values in the

Freezing vector. Display a message first to make your output more readable.
```
  Number of readings below freezing:
  ans =
     208
```

6.8. Longest duration between freezes

The **Freezing** vector tells you the locations in the **Temps** vector where the temperature is at or below freezing.

The first 16 columns of the **Freezing** vector look like this:
120 121 122 123 124 125 126 127 128 129 130 131 195 196 197 198
This means that readings 120 through 131 were all below freezing, then notice that the temperature went above freezing until reading number 195 when it went below freezing again. The number of readings between 195 and 131 is 195 - 131 - 1 = 63, so there was a duration of 63 readings when the temperature was above freezing.

Calculate the longest duration between freezing temperatures in the vector.

To do this you must subtract each element in the **Freezing** vector from the element that comes after it:
121 - 120 - 1 = 0
122 - 121 - 1 = 0
...
130 - 129 - 1 = 0
131 - 130 - 1 = 0
195 - 131 - 1 = 63
196 - 195 - 1 = 0
...
All those differences must be stored in a new vector that will look like this:
0 0 0 0 0 0 0 0 0 0 0 63 0 0 0 ...

The way to construct that vector is to use two vector ranges. After you have created the **Freezing** vector, observe what happens when you type this into Matlab:
```
  EDU>> Freezing(2:end)
  ans =
    Columns 1 through 10
     121   122   123   124   125   126   127   128   129   130
    Columns 11 through 20
     ...
    Columns 201 through 207
     745   755   757   758   761   762   763
```
The **Freezing** vector has 208 elements, but that vector (the answer that Matlab just showed you) has only 207 elements. Likewise, the following vector also has 207 elements:
```
  EDU>> Freezing(1:end-1)
  ans =
    Columns 1 through 10
     120   121   122   123   124   125   126   127   128   129
    Columns 11 through 20
     ...
    Columns 201 through 207
     744   745   755   757   758   761   762
```
If you subtract those two vectors, then subtract 1 from *that* vector, you will end up with a vector that shows how many readings were above freezing:
```
  ans =
    Columns 1 through 10
      0     0     0     0     0     0     0     0     0     0
    Columns 11 through 20
      0   63    0    0    0    0    0    0    0    0
    Columns 21 through 30
      0   94    0    0    0    0    0    0    0    0
    Columns 31 through 40
      0    0   13    0    0    0    0    0    0    0
    Columns 41 through 50
      ...
    Columns 201 through 207
      0    9    1    0    2    0    0
```

You can see that in the 12th column there is the number 63. That means that at that point in the temperature readings, there were 63 consecutive readings above freezing.

Find the maximum number of consecutive readings that were above freezing and display the number with a message before it, like this:

```
  Maximum consecutive readings above freezing:
  ans =
     112
```

6.9. Determine mean number of consecutive readings above freezing

Find the mean value of the vector you created in the previous step. Do not include any zeros in this calculation. Create a new vector of only the non-zero values from that vector. It should start like this:

```
  63  94  13   8  13  18  10 ...
```

This is not as hard as it sounds. Consider how the **find()** function works. It's designed to do something just like this. The problem is that **find()** tells you locations, whereas you want the values at those locations. So after you find the locations where that vector is not zero, you need to get the values at those locations. It turns out that you already know how to get the value of a vector at a location, or a bunch of values at a bunch of locations.

Then display the mean of that vector with a message:

```
  Mean consecutive readings above freezing:
  ans =
     24.2222
```

6.10. Plot the temperatures

Plot the values in the Temps vector. Do this by typing this statement into the Matlab command prompt. It does not have to be in your program.

```
  >> plot(Temps)
```

Save the plot to a file so that you can use it to write your document. You do not need to label the plot in any way.

To save the plot, click the blue **Save Figure** icon shaped like a floppy disk in the plot window.

Save the plot as type **Portable Network Graphics (PNG)**, not as the default FIG. Microsoft Word does not know how to import FIG files.

7. Deliverables

Create a Microsoft Word or OpenOffice Word file (or some other format that your TA agrees on -- ask him or her if you are not sure). Save the file with the name **Project2** with a .doc or .docx format.

At the beginning of the document include this information:

Analysis of Temperature Data
CSE1010 Project 2, Fall 2011
Your name goes here
The current date goes here
TA: *Your TA's name goes here*
Section: *Your section number goes here*
Instructor: Jeffrey A. Meunier

Be sure to replace the parts that are underlined above.

Now create the following five sections in your document.
1. Introduction
In this section copy & paste the text from the introduction section of this assignment. (It's not plagiarism if you have permission to copy something. I give you permission.)
2. Data
Copy & paste the allTemps function and write one sentence before it describing what the data represents.
3. Output
Run your program and copy & paste the output from the command window here. You do not need to write anything.
4. Graphs
Insert the plot in this section. To do this, you can drag & drop the file from wherever you saved it right into MS Word. Just indicate in the MS Word document that it is a plot of the temperature data. Do not submit the plot image, but don't delete it until after your assignment has been graded. Actually, don't delete any file that you create for this course. Not ever, except maybe after you graduate.
5. Source code
Copy & paste the contents of your main.m file here. You do not need to write anything.

8. Notes

- This is not a long program, but you will need to do some thinking and experimenting in Matlab. Use the book and your notes.
- Make sure your program displays these 6 values:
    1. Maximum temperature
    2. Minimum temperature
    3. Mean temperature
    4. Number of readings below freezing
    5. Maximum consecutive readings above freezing
    6. Mean consecutive readings above freezing

9. Submission

Submit two things on HuskyCT:
1. Your **main.m** file.
2. Your **Project2** MS Word document.

If for some reason you are not able to submit your files, email your TA before the deadline. Attach your files to the email.

10. Grading rubric

Your TA will grade your assignment based on these criteria:

- (2 points) The comment block at the beginning of the program file is correct.
- (6 points) The program displays the correct answers.
- (4 points) The program uses the correct calculations to generate the answers.
- (4 points) The program is formatted neatly. Follow any example in the book, or see the web site here: https://sites.google.com/site/jeffscourses/cse1010/matlab-program-formatting
- (4 points) The document contains all the correct information and is formatted neatly.

11. Getting help

Start your project early, because you will probably not be able to get help in the last few hours before the project is due.

- If you need help, send e-mail to your TA immediately.
- Go to the office hours for (preferably) your TA or any other TA. I suggest you seeing your own TA first because your TA will know you better. However, don't let that stop you from seeing any TA for help.
- Send e-mail to Jeff.
- Go to Jeff's office hours.