Texas A&M University
ENGR 112 - Spring 2017

MATLAB Project

Introduction.

This project involves the development of a MATLAB-based, menu-driven application to aid in the statistical analysis of numerical data. This is an individual project, but you are encouraged to work together on the required programming concepts. Each student will be responsible for submitting their own deliverables. (See below.)

Basic statistical analysis is part of the curriculum for ENGR 112. Each time we learn a new analysis technique, we will follow up by demonstrating how to use MATLAB to implement that technique. So, for example, when we discuss the definition of terms such as *mean*, *median*, and *mode*, we will demonstrate how to use MATLAB's built-in functions and/or other programming techniques to calculate numerical values associated with these definitions for a particular data set. The expectation is that after learning a new technique, you can simply add new functionality to your code. In other words, you will develop this code in "chunks" which you can fully debug before moving on to the next technique.

Use of the Application.

The scenario for using the code is that someone has sent the user a text file of numerical data and has asked the user to provide a report listing various statistical quantities associated with the data. The data format will be one or two columns of data with an unknown number of data elements. The basic descriptive statistical quantities (*mean, median, mode, variance, standard deviation, range*) are easily calculated using MATLAB built-in functions and their values can be written to a summary file.

Next, the user should determine whether or not the data is "normally distributed." The user can determine this by using various plots such as `histogram` or `normplot` within MATLAB. If, in the user's judgment, the data is normally distributed, the user is allowed to perform z-table calculations resulting in the output of probabilities or intervals of x or z.

General Features.

The following list represents the general features of the program:

- The program should be menu-driven.
  - We will demonstrate in-class how to build a simple text-based menu.
- Input will consist of the name of an existing data file consisting of one or two columns of an unknown number of data elements.
- All output and results will be directed to both the screen and an output file.
  - Output filename should be chosen by user.
  - The data filename should be written to the output file along with date, user name, etc.
  - Values for all the descriptive statistics listed below should be calculated and written to the output file in the following format:
    - Mean = XXX.YY

        - Median = XXX.YY
        - Mode = XXX.YY
        - Var = XXX.YY
        - Stdev = XXX.YY
        - Min = XXX.YY
        - Max = XXX.YY
        - Count = XXXXXX
    - Please note that the equal signs and decimal points are aligned vertically.
    - Please note that since Count is an integer, it has no decimal point.
    - Note that if Count > 30, use the population standard deviation and put an extra message in the output. For Count <= 30, use sample standard deviation and put an extra message in the output.
    - Other output associated with z-table questions and answers should also be written to the output file.

Text-Based Menu.

The following is a list of possible menu choices. You have the freedom to design the user interface in a manner you see fit.

- Menu Choices (You may choose a different menu scheme.)
    - Set user name
    - Load data file
    - Clear data from memory
    - Set output filename
    - Plot histogram
    - Plot histogram fit
    - Plot probability plots
    - Regression of y on x
    - Find probability given x or z
    - Find x or z given probability
    - Exit

Error Conditions.

It is important for a computer application to give the user guidance with regard to errors and to fail gracefully in the event that unrecoverable errors occur. The following is a list of possible error conditions that your code should check and attempt to correct (possibly with additional user input).

- Error/Warning Conditions (Examples)
    - Trying to load another file when data is in memory
    - Before using z tables, user must answer question: “In your judgment, is the data normally distributed?” The answer should be written to the output file.
    - Input file does not exist.
    - Output file already exists. Data will be overwritten.
    - Invalid probability entered by user. (If negative or greater than one.)
    - Invalid x or z entered by user.

    - For regression, input vectors of unequal length
    - Invalid menu selection entered.

Coding Standard.

It is important to observe good coding practices when implementing large code projects, like this one. The following is a short list of guidelines that you should follow when writing your code. It is by no means an exhaustive set of guidelines for creating high quality code, but it is definitely the minimum requirement.

- Comment your code
    - Comments should explain the code's intent or summarize what it does
    - Comments should be kept up to date
    - Comments should be clear and correct
    - Avoid endline comments
    - Focus on why rather than how
    - Avoid abbreviations
    - Remove extraneous, redundant, and self-indulgent comments
    - Comment on units and ranges of data
    - Comment on limitations of input data
    - Comment every global variable
    - Don't use a comment where renaming a variable or function would also work
    - Explain the purpose of every routine and give facts about its input, output, usage, limitations, error corrections, global effects, and sources of algorithms
    - Every file should have the standard ENGR 111/112 header
    - **Good code should document itself; if the code requires extensive commenting, rewrite the code**
- Variable and routine naming
    - Names should fully and accurately describe the entity the variable represents, or the function the routine computes
    - Names should not be too short or too long, somewhere between 10 and 16 characters on average is best
    - Pick a naming convention and use it consistently
        - Differentiate between variable names and routine names, between global and local variables, between constants and variables
        - Use camel case, underscoring, or both (for different things), but not neither
            - Camel Case: numberOfWidgets
            - Underscoring: number_of_widgets
        - Constants are all uppercase: GRAVITY, COEFF_FRICTION, KG_PER_LBM
        - Indices are i,j,k
        - c is a character, s is a string, n is a counting number
- Error handling
    - Practice defensive programming
    - Handle garbage in

        - Check values of data from external sources
        - Check values of all input parameters
        - Decide how to handle bad inputs
    - Handle expected errors appropriately, e.g. one or a combination of
        - Substitute the next piece of valid data
        - Substitute the closest legal value
        - Return the same answer as the previous time
        - Return an error code
        - Display an error/warning message
        - Log an error/warning message to a file
        - Shut down
    - Fail gracefully on unexpected errors:
        - use try/catch blocks
        - make it very hard to crash the program
- Code layout and style
    - Use whitespace appropriately to maximize readability
        - Spaces, tabs, newlines
    - Good visual layout accurately and consistently represents the logical structure of a program
    - Group related code together into blocks, like paragraphs in an essay
        - Put a blank line between blocks
    - No more than 80 characters per line
        - Indent continuation lines

Deliverables.

As stated above, this is an individual project and each student is responsible for the deliverables listed below.

There will be two deliverables for this project:

1. Your MATLAB code. We will need all *.m files required to run your code including any user-defined functions that you may use.
2. A short report describing the operation of your code (i.e. a user manual, similar to MATLAB's own help documentation) and how you went about testing that the code works properly and provides correct results. The testing section of your report should state what tests the code has to pass, argue that the tests adequately cover the specifications for the program, and demonstrate that the code passes the tests. This report should be typed in 12pt Times New Roman with 1-inch margins and double spaced lines. The report should not be more than 5 pages in length.

Please note: The above represents minimal requirements for the project. You are encouraged to add additional features to make the program more useable and useful for you. If you do so, the teaching team applauds your effort and is happy to help you.

Grading.

The grade for this project is worth 6% of your total course grade as stated in the syllabus. The rubric will be made available on the eCampus site.

As part of the grading procedure for this project, all student code will be submitted to the MOSS system to search for examples of plagiarism. MOSS (Measure of Software Similarity) can detect the level of similarity between any one code and all others submitted. Software plagiarism is considered cheating under the Aggie Honor Code. So, please work together at the concept level, but write your own code for submission.