|  |
| --- |
| [Company name] |
| MATLAB Project |
| Statistical Analysis Project Development Report |

|  |
| --- |
| Brandell Ellison  5-1-2016 |

# OBJECTIVES STATEMENT

# Style Guide

# Program Description

# Foundational Paradigms

# Special notes

The program will not support regex (regular expression) based text editing.

# %% GENERAL CHECKLIST:

% Open or create a user defined file

% Edit file into a usable state

% Save and close edited file

% Load file

% Open to main menu

% Process data and generate a statistical report

% Publish results in a pdf format

% Close

Include the following:

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.

# SPECIFICations

The program should be menu-driven.

o 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.

o Output filename should be chosen by user.

o The data filename should be written to the output file along with date, user name, etc.

o 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

o Please note that the equal signs and decimal points are aligned vertically.

o Please note that since Count is an integer, it has no decimal point.

o 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.

o Other output associated with z-table questions and answers should also be written to

the output file.

%% UI DESIGN:

ext-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.

o Menu Choices (You may choose a different menu scheme.)

+ o Set user name

>> o Load data file

+ o Clear data from memory

o Set output filename

o Plot histogram

o Plot histogram fit

o Plot probability plots

o Regression of y on x

o Find probability given x or z

o Find x or z given probability

o Exit

%% DEFENSIVE PROGRAMMING MEASURES:

% Error/Warning Conditions (Examples)

o Trying to load another file when data is in memory

o 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.

o Input file does not exist.

o Output file already exists. Data will be overwritten.

o Invalid probability entered by user. (If negative or greater than one.)

o Invalid x or z entered by user.

o For regression, input vectors of unequal length

o Invalid menu selection entered.

%% FORMATTING AND STYLE GUIDE:

• Comment your code

o Comments should explain the code’s intent or summarize what it does

o Comments should be kept up to date

o Comments should be clear and correct

o Avoid endline comments

o Focus on why rather than how

o Avoid abbreviations

o Remove extraneous, redundant, and self-indulgent comments

o Comment on units and ranges of data

o Comment on limitations of input data

o Comment every global variable

o Don’t use a comment where renaming a variable or function would also work

o Explain the purpose of every routine and give facts about its input, output, usage,

limitations, error corrections, global effects, and sources of algorithms

o Every file should have the standard ENGR 111/112 header

o Good code should document itself; if the code requires extensive commenting,

rewrite the code

• Variable and routine naming

o Names should fully and accurately describe the entity the variable represents, or the

function the routine computes

o Names should not be too short or too long, somewhere between 10 and 16 characters

on average is best

o 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

o Practice defensive programming

o Handle garbage in

? Check values of data from external sources

? Check values of all input parameters

? Decide how to handle bad inputs

o 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

o Fail gracefully on unexpected errors:

? use try/catch blocks

? make it very hard to crash the program

• Code layout and style

o Use whitespace appropriately to maximize readability

? Spaces, tabs, newlines

o Good visual layout accurately and consistently represents the logical structure of a

program

o Group related code together into blocks, like paragraphs in an essay

? Put a blank line between blocks

o No more than 80 characters per line

? Indent continuation line

function [] = print\_menu\_options()

%% displays main menu options.

clc;

i=0;

while i == 0

% Display the menu to the user

%print\_menu\_options;

fprintf('What do you want to do?\n');

fprintf('1) Set Username\n');

fprintf('2) Load Data File\n');

fprintf('3) Clear Memory\n');

fprintf('4) Set Output Filename\n');

fprintf('5) Plot Histogram\n');

fprintf('6) Plot Histogram Fit\n');

fprintf('7) Plot Probability Plots\n');

fprintf('8) Regression of y upon x\n');

fprintf('9) Find Probablilty given x or z values\n');

fprintf('10) Quit\n');

% Print prompt

fprintf('Enter your choice: ');

% Get user input

% For safety, read as a string to prevent MATLAB from evaluating input

choice = input('','s');

% Determine which action to take based on the user's input

switch choice

case '1'

Set\_Username;

case '2'

%Load\_Data

do\_fun\_stuff

case '3'

%Clear\_Mem

do\_useful\_stuff

case '4'

%Set\_Out\_File

do\_boring\_stuff

case '5'

%Hist\_Plot

do\_fun\_stuff

case '6'

%Hist\_Fit

do\_useful\_stuff

case '7'

%Prob\_Plots

case '8'

%load data here

%Regression

do\_fun\_stuff

%additional action here

case '9'

%Find Probability

do\_useful\_stuff

case '10'

%Quit

%continueProgram = false;

do\_boring\_stuff

i=1;

otherwise

% This case handles invalid input

fprintf('\nERROR: Please enter an integer between 1 and 10\n\n');

% Wait for user to acknowledge the error message

fprintf('Press any key to continue');

pause;

end % end of switch

end

end

# Conceptual Problems:

Ideal Trajectory.:

1. Run main menu()
   1. Switch a triggered
      1. Case a selected
         1. %case a executes
      2. %switch.case.A ends; return
   2. Switch ends.
2. Return to main menu
   1. End

# Independent functions

## Menu:

ContinueProgram = true

I = 0

While continueprogram==true

A = menu('What do you want to do?','Set Username','Load Data File','Clear Memory','Set Output Filename','Plot Histogram','Plot Histogram Fit','Plot Probability Plots','Regression of y upon x','Find Probablilty given x or z values','Quit');

switch a

case a == 1

case a == 2

case a == 3

case a == 4

case a == 5

case a == 6

case a == 7

case a == 8

case a == 9

case a == 10

end

end

## function [ID\_User] = Set\_Username

%% Sets user name as a global variable

% Input: string of alphanumeric values

% Output: A global variable that carries the userid

% the string throughout the program.

clc;

ID\_User = input('Enter your user name.', 's');

global ID\_User;

fprintf('\nWelcome, %s !\n',ID\_User)

fprintf('Press any key to continue');

pause;

% >> regex is to be used to check contents of the string, if time permits.<<

%%TLDR: CHARACTERS THAT ARE NOT ALLOWED BY THE fprintf() FUNCTION ARE NOT ALLOWED TO BE IN A USERNAME.

%{

if isstring(ID\_User) == 0

error('Username must be a string.')

elseif % contains numbers

error('Names can only contain alphabetical characters,\nhyphens, and apostrophes.')

elseif% contains special characters except for hyphens and apostrophes.

error('Names can only contain alphabetical characters,\nhyphens, and apostrophes.')

elseif% does not use English characters

error('Names can only contain alphabetical characters,\nhyphens, and apostrophes common to the English language.')

%otherwise

%Pass through.

%}?

%}

1. end

## function [ ] = Load\_Data

%% Load file data

x=uigetfile(‘\*.\*’,‘Identify the file you would like to work with.’,’Placeholder.mat’)

(Check file and data for error and other such entropy.)

end

## function [ ] = Clear\_Memory

%% Clear\_Memory()

% Gets rid of all data in current workspace

%

%

%

%

clear\_data=menu('Are you absolutely sure you want to do that? Once deleted, your data cannot be recovered.’,’Yes’,’No’)

switch clear\_data

case 1

clear,clc,close all

menu('Memory has been successfully cleared',’Accept’);

case 2

return

end

## function [ ] = Set Output Filename ()

end

## function [ ] = Plot Histogram (data)

end

## function [ ] = Histogram Fit (data)

end

## function [ ] = Probability Plot (Data)

[description]

load slump.txt

x=70:.1:130;

a=mean(slump);

b=std(slump);

c=normpdf(x,a,b);

d=normcdf(x,a,b);

e=max(size(slump));

z=1.64;

u\_conf= a+z\*(b/(e)^.5);

lo\_conf= a-z\*(b/(e)^.5);

subplot(1,2,1)

plot(x,c,'r')

title('pdf')

xlabel('slump (mm)')

ylabel('probability')

axis([70,130,0,0.05])

subplot(1,2,2)

plot(x,d,'b')

title('cdf')

xlabel('slump (mm)')

ylabel('cumulative probability')

axis([70,130,0,1])

sprintf('The mean slump is %.4g mm, and the standard deviation is %.3g mm.\n\n Based on %0.1f samples, a %2.1f%% confidence interval for the average slump \n of this concrete mix is between %.2f mm and %.2f mm \n',a,b,e,90,lo\_conf,u\_conf)

z = pdf(pd,x)

figure

plot(x,z)

end

## function [ ] = Regression (y,x)

P=[0.6,1,1.4,1.8,2.2,2.6,3,3.4,3.8,4.2,4.6,5,5.4];

K=[5.89,4.68,4.1,3.45,3.25,2.69,2.25,1.92,1.55,1.45,1.19,0.99,0.84];

k=log(K);

a=polyfit(P,k,1);

B=a(1);

A=exp(a(2));

Kfit=A\*exp(B\*P);

e=(Kfit-K);

MSE=mean(e.^2);

fprintf('The MSE of each regression is listed respectively:\n')

disp(MSE.')

%

a2=polyfit(P,K,1);

B2=a2(1);

A2=a2(2);

Kfit2= polyval(a2,P);

e2=(Kfit2-K);

MSE2=mean(e2.^2);

fprintf('The MSE of each regression is listed respectively:\n')

disp(MSE2.')

i=1;

comparison= MSE2 > MSE;

fprintf('Order\tMSE\n');

%

i=0;

while comparison == 1

i=i+1;

a2=polyfit(P,K,i);

B2=a2(1);

A2=a2(2);

Kfit2 = polyval(a2,P);

e2 = (Kfit2-K);

MSE2 = mean(e2.^2);

comparison = MSE2 > MSE;

attempt(i)=i';

Meanerrsqr(i)=MSE2';

end

table(attempt',Meanerrsqr','VariableNames',{'Order','MSE'})

fprintf('Polynomial regression of order %1.0f has a better \n MSE than log transformed linear regression.\n\n',i)

%return to main menu

Modify the above code.

end

## function [ ] = Probability Fndr (data,x)

Norm pdf norm cdf, normplot,

probplot(y)

end

## function [ ] = Quit ()

while continueprogram = true

quit = menu(‘Are you sure you want to quit? Any unsaved data will be lost.’,’Yes’,’No’)

switch quit

case quit == 1

menu(‘Matlab is shutting down.’,’Accept’)

continueProgram = false

case quit == 2

return

end

## Demonstration code

### Gaussian prob plot demonstrating potential for excel input

clear,clc,close all,format bank

x=double.empty

[fn,pn,fi]=uigetfile({'\*.mat';'\*.m';'\*.xlsx';'\*.txt';'\*.\*'},'Select list','Normal.txt');

if isstruct(fn) == 1 ;

x=fn.x

fnx(:,1) = fn.y;

elseif fi == 3; %is an excel file

fnx=xlsread(fn,-1);

else isstruct(fn) == 0;

fnx=double(fn); % converts from var to array

end

[row,column] = size(fnx);

if row < column;

fnx = fnx'

end

C=userstat(fnx);

if isempty(x) == 1;

x=linspace(0,C(9,1),C(10,1));

end

PDF = normpdf(x, C(1,1), C(6,1));

CDF = normcdf(x, C(1,1), C(6,1));

figure

subplot(1,2,1)

plot(x,PDF)

xlabel('Criteria (Units)')

ylabel('Probability')

%axis([min(x),max(x),min(PDF),max(PDF)])

subplot(1,2,2)

ylabel('Cumulative Probability (Decimal)')

plot(x,CDF)

%axis([min(x),max(x),min(CDF),1])