Computing for Scientists - Lab 2

CS 1340 — Dr. Mihail
Department of Computer Science
Valdosta State University

September 4, 2018

1 Introduction

In this lab, you will learn how to create a script, learn about MATLAB functions and syntax, and practice using some of the built-in functions. **Problems 1 and 2 are due before the lab period ends, on Thursday, August 30th**.

2 Create a script

Once you open MATLAB, clicking on the HOME tab, and the New Script button, will open the Editor window, with an untitled, blank document. Here you will write MATLAB *scripts*, which are simply an ordered set of commands, that when ran (using F5 or the EDITOR tab Run button) will be executed line by line, just as you would normally type them in the command window. Scripts can be saved for later use, and executed any number of times. Scripts have the .m extension.

The following script prompts the user for a value, then computes the square of that value and prints the results.

```
% CS 1340 Lab 2 Problem 1
% author: Dr. Mihail
% purpose: simple MATLAB script that squares a user's numeric input
x = input('Please enter x: ');
fprintf('You entered %.0f\n', x);
xSquared = x^2;
fprintf('%.0f squared is %.0f\n', x, xSquared);
```

Below is a sample interaction, when the script above named `test.m` is ran:

```
 >> test
Please enter x: 10
You entered 10
10 squared is 100
>>
```

Problem 1 (25 points) Modify this script such that it prints the number you entered with 2 significant digits (currently displays 0 significant digits). Submit your script on the course electronic assignment submission page. You may name this script `test.m`, however, please make sure to select lab assignment 2, and change the prolog with your name and change the purpose to "solution to problem 1".

A sample interaction with a correctly modified program is:

```
>> test
Please enter x: 10.45
You entered 10.45
10.45 squared is 109.20
>>
```

The submission is due before the lab period ends, Thursday, January 29th.

Problem 2 (25 points) Create a script that prompts the user for the two sides of a right triangle: a and b. Your script will compute the length of the hypotenuse, which is: $c = \sqrt{a^2 + b^2}$. A sample interaction with the program is as follows:

```
>> triangle
Please the length of side 1: 3
Please enter the length of side 2: 4
The hypotenuse of the triangle, with 2 significant digit precision is: 5.00
>>
```

Please ensure you have a prolog and the prompts and output text are **exactly** the same as in the sample above.

Submit this script using the course assignment web page. The script is due before the end of the lab period. Make sure you have a prolog.

3 Functions

As you have seen in the Math Review slides, functions should be thought of as a set of operations that take place in a *black box*. Functions take input, called **arguments** and produce output. There can be any number (including none) of arguments. There can also be any number (or none) of outputs. The syntax used in MATLAB to **call** a function is:

```
[o1, o2, ...] = function_name(arg1, arg2, arg3, ...)
```

For example, in the following interaction from the command window, we create a matrix A, and **call** the function *size*:

```
>> A = [1 2 3; 3 4 5]

A =
```

```
     1     2     3
     3     4     5

>> [rows, columns] = size(A)

rows =

     2


columns =

     3

>>
```

The variables `rows` and `columns` have been created as a result of the function call.

MATLAB has several built-in functions. On Page 60 in your textbook, there are a few examples of elementary mathematical function descriptions. For example, `abs(x)` computes the absolute value of `x`. Please read over the others, e.g.: `sqrt, round, floor, ceil,` and `fix, sign, exp, log, log10, log2`. These functions take one argument. Others take two arguments, for example `rem(x, y)` returns the remainder of the division x divided by y.

4 Using MATLAB help

MATLAB documentation can easily be accessed using the `doc` command, for example `>> doc sqrt` will open a help window, with more information about the `sqrt` function. The `doc` command is extremely useful and you will use it througout the semester.

5 MATLAB functions to create matrices

It is often the case that you know the size of a matrix and you wish to create it. MATLAB provides several functions to create matrices (and vectors) with all elements set to a default value. For example, the function `ones(r, c)` creates a matrix with r rows and c columns, all elements set to 1. For example, to create a ones matrix with all elements A with 3 rows and 4 columns:

```
 >> A = ones(3, 4)

A =

     1     1     1     1
     1     1     1     1
     1     1     1     1
```

Similarly, the function `zeros(r, c)` that creates a matrix of size r by c, with all 0's.

```
 >> B = zeros(3, 4)

B =

     0     0     0     0
     0     0     0     0
     0     0     0     0
```

A square identity matrix, can be created with the function `eye(s)`, where s is the size of the square matrix:

```
 >> I = eye(4)

I =

     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
```

A random matrix can be generated using the `rand(s)`, where s is the size of the square matrix. The numbers are uniformly distributed in the range $[0, 1]$ for example:

```
>> A = rand(5)

A =

    0.8147    0.0975    0.1576    0.1419    0.6557
    0.9058    0.2785    0.9706    0.4218    0.0357
    0.1270    0.5469    0.9572    0.9157    0.8491
    0.9134    0.9575    0.4854    0.7922    0.9340
    0.6324    0.9649    0.8003    0.9595    0.6787
```

You can extract a subset of the matrix using *slicing*. Slicing syntax involves use of colon (:) as follows:

- start:end - will generate a column vector from *start* to *end* including *start* and *end*. Example: 1:4 will result in $[1, 2, 3, 4]$.
- start:step:end - will generate a column vector from *start* to *end* with step *step*. Example: 1:2:10 will result in $[1, 3, 5, 7, 9]$.
- : - when used in matrix indexing, everything (columns or rows) are returned.

We can use slicing to extract a column from a matrix. For example, if we wanted to extract the second column from matrix A above, we would enter:

```
>> A(:, 2)

ans =

    0.0975
    0.2785
    0.5469
    0.9575
    0.9649
```

When slicing, `end` is a keyword that can be used to refer to the index of the *last* column or row. For example, to get the last column we could use the following slicing operation:

```
>> A(:, end)

ans =

    0.6557
    0.0357
    0.8491
    0.9340
    0.6787
```

To slice the last row:

```
>> A(end, :)

ans =

    0.6324    0.9649    0.8003    0.9595    0.6787
```

To slice subset of rows and columns, for example:

```
 >> A(1:3, 1:3)

ans =

    0.8147    0.0975    0.1576
    0.9058    0.2785    0.9706
    0.1270    0.5469    0.9572
```

or

```
>> A(end-2:end, end-2:end)

ans =

    0.9572    0.9157    0.8491
    0.4854    0.7922    0.9340
    0.8003    0.9595    0.6787
```

Problem 3 (Homework) (50 points) In a fictitious college introductory computer science class of 1000 students, the professor sampled each student's age (months, days, hours, minutes, and seconds turned into decimal part of a year), height in inches and SAT score. The data is available as an Excel spreadsheet at:
`http://mypages.valdosta.edu/rpmihail/teaching/S17/CS1340/datasets/lab2/age_height_sat.xlsx`.

Download the file and save it. You will use this file in MATLAB as your dataset. Write a script that reads the Excel file into a matrix using the function `xlsread`. It requires one argument, the filename, of type string (text surrounded by single quotes). For more information, type `doc xlsread` in the command window. For example:
`data = xlsread('age_height_sat.xlsx');`
will create a matrix named `data` from the Excel spreadsheet. To ensure it is properly read, check the size of *data* using the `size` function:

```
>> size(data)

ans =

        1000           3
```

In the script, write code to compute (and display using `fprintf`) the minimum, maximum, median and mean age, height and sat score. The MATLAB functions to compute these *descriptive statistics* are `min, max, median, mean`. You will have to extract columns from the dataset to report these statistics.

The standard deviation σ is a descriptive statistic that gives a measure of data "spread" away from the mean. For a vector x (size N), the standard deviation is computed using the following formula:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2} \tag{1}$$

Write MATLAB code (in the same script) to compute the standard deviation for age, height and SAT score. You **cannot** use the MATLAB function `var`.

The distribution of samples (age, height and SAT score) can be visualized using a *histogram*. Use MATLAB `hist` to generate a histogram with both 20 and 40 bins for age, height and SAT. The function will produce a figure, which you can save as an image (File→Save As→Type .jpg or .png). For the three variables, you will have 6 histograms. You will import them into a Microsoft Word document that you submit for Problem 3. Please carefully annotate (label each Figure image, explaining what it represents). You will also submit the script you wrote for Problem 3.

Due dates

1. Lab Problems 1 and 2 are due at the end of the lab period, 8/30/2018.
2. Lab Problem 3, due by Wednesday before midnight 09/05/2018. For Problem 3, you have to submit two files, a MATLAB script .m file and a .docx file.