PSYC40012 Models of Psychological Processes

Tutorial Video 2

In this video, I will show in detail how to manipulate variables that you have created and stored in the workspace. I will also show you how to load variables stored in files outside of Matlab and save variables from Matlab to external files (which you can load into other software packages such as Excel or SPSS).

First, let's discuss how you manipulate matrices and retrieve information about the entries in those matrices.

Getting the size of a matrix or vector

Let's start by creating a matrix:

```
A = [6 5; 4 3; 2 1]
size(A) % returns the size of the matrix
```

The variable returned from the function size is a 1 x 2 matrix. (To think about: How many rows and columns does this variable have?)

```
sz = size(A) % save the size to a variable called sz
size(A,1) % size of first dimension; number of rows
size(A,2) % size of second dimension; number of cols
```

For vectors, we can still use the size command but there are other commands which are useful as well. First, create a vector:

```
v = [6 2 8 2]
length(v) = 4; % length of longer dimension
length(A) = 3; % length of longer dimension
```

Since length returns the length of the longer dimension, it is usually better to use length only for vectors and to use size for matrices.

```
numel(v) = 4; % total number of elements
numel(A) = 6; % total number of elements
```

Accessing the elements of a matrix of vector

Matlab uses the round brackets (parentheses) to index elements of a matrix

```
A = [6 5; 4 3; 2 1]

A(3,2)

ans =

          1

% returns the entry in the 3rd row, 2nd column of X
```

Remember from the first video that we always lists rows before columns. Another way of stating this is that in Matlab, the first dimension is the rows and the second dimension is the columns.

```
A(2,:)
```

The colon means return everything in that row/column, so `A(2,:)` means return everything in the 2nd row

(To think about: How would we return just the first column?)

```
A([1 3], :)
```

We can use vectors as indexes to the elements of a matrix. This statement returns everything from the 1st and 3rd rows of the matrix A. This is a more sophisticated method of indexing.

Logical indexing

Just as we can use a vector to index the rows or columns of a matrix, we can also use logical operations to index the elements of a matrix.

```
A = [6 5; 4 3; 2 1]

A(A(:,1) > 3, :) % returns any row in which the element
in the first column is greater than 3

A(A <=4) % returns any element less than or equal to four
(Note that the structure of A is not maintained by this
command)
```

Assigning variables

We can also use indexing to change the variables of a matrix:

```
A(1,2) = 7 % changes the element in the 1st row and the 2nd column to the value 7

A(:,2) = [10; 11; 12] % changes the 2nd column
```

We can append a new matrix to an existing matrix

```
A = [A, [100; 101; 102]] % Append another column on the right of the A matrix

A(:) % put all elements into a single column vector


A = [1 2; 3 4; 5 6]

B = [11 12; 13 14; 15 16]
```

Concatenate a matrix

```
C = [A, B] – puts matrices A and B together

C = [A; B] – puts matrices A and B together but puts B at the bottom
```

Loading and saving data

In the last video, we simulated some IQ values with a mean of 100 and a standard deviation of 10 using the equation:

```
IQ = 100 + 10 * randn(1, 10000)
```

Imagine now that you had collected this data, and you have stored these variables in a text file called IQ.txt. (It's probably easiest to view this file in a program like WordPad; Notepad will work as well, but will probably not be as nice to look at).

We can read this data into Matlab using the following command:

```
load IQ.txt

load('IQ.Txt')
```

To see that the variable has loaded we can look at the workspace or type:

```
who
```

```
IQ

size(IQ)

clear IQ % removes the variable from the workspace

clear all % removes everything from the workspace


load IQ.txt
```

There are other methods to read data into Matlab. To read a delmited text file, we could use:

```
IQ = dlmread('IQ.txt')
```

which reads the delimited txt file and assigns the content to the variable `IQ.`

(Load and dlmread work when the text file contains variables which all the same type (e.g., all numbers). There are more powerful commands for reading in files with mixed types such as `textread` or `textscan`. For these commands, you need to specify the type of each variable (e.g., string, number, and so on. This can get a bit complicated). You can also use `csvread` or `xlsread` to read comma separated csv files or even Excel spreadsheets.)

Once you've read your data into Matlab, then you can manipulate that data.

```
x = IQ(1:10, 1) % Sets x to the first 10 rows of the
first column of the data matrix

save mydata.mat x;
```

This function saves a .mat file containing the variable x to your computer. The .mat files can be read easily by Matlab, and I can load this variable by typing

```
load mydata.mat
```

To write this output to a file, I can use:

```
save mydata.txt x –ascii % save to a text file
```

Or

```
dlmwrite('mydata.txt', x) % save to a text file
```

In the next video, we will discuss using functions to perform operations on variables.