Statistics and Data Analysis in MATLAB
Kendrick Kay, kendrick.kay@wustl.edu

MATLAB Basics II

1. Figures and plotting

`figure` - create a figure window
`hold on` - hold figure so that new plot elements will add to (not replace) existing elements
`plot` - draw lines
`scatter` - draw a scatter plot
`bar` - draw bar chart
`hist` - draw histogram
`image`, **`imagesc`** - draw a 2D matrix as an image
`axis` - obtain or change axis bounds
`xlabel`, **`ylabel`** - add a label to the x- or y-axis
`title` - add a title
`legend` - add a legend
`colormap` - obtain or change the colormap
`colorbar` - add a colorbar
`get` - get properties of a plot element
`set` - set properties of a plot element
`gcf` - get handle of current figure
`gca` - get handle of current axis
`print` - print figure to an image file (e.g. PNG, EPS)
`close` - close figure window
`subplot` - split a figure into multiple smaller plots

2. Flow control

`if` ... **`else`** ... **`end`** - if something is true, do this; otherwise, do that
`while` ... **`end`** - while some condition is true, repeat statements
`for` ... **`end`** - for each iteration, repeat statements
`switch` ... **`case`** ... **`end`** - execute different statements depending on value of a variable
`break` - exit a for- or while-loop prematurely
`continue` - jump to the next iteration in a for- or while-loop

3. Boolean operators

`&&` - if something is true AND something is true
`||` - if something is true OR something is true
`~` - NOT
`all` - return true if all elements are true
`any` - return true if any element is true

4. Path issues

- The **path** is a collection of directories within which MATLAB searches for .m files. In order for MATLAB to recognize the existence of a function (or script), that function (or script) must exist on the path.
- The current working directory is automatically included in the path.

- If there are multiple .m files that share the same name, one of the files takes precedence. To determine which file has precedence, use the `which` command.

5. Path-related commands

`path` - get the current path
`addpath` - add to the path
`genpath` - create a path string from a directory, including all sub-directories

6. Writing functions

- Writing good functions is extremely important; if done well, programming complex software can become easy.
- A good function is general (i.e. it can be re-used for different situations) and well-documented (i.e. the inputs and outputs of the function are clearly described).
- To add comments, use the `%` symbol. Text following the `%` symbol is ignored by MATLAB.
- Simple example of a function:

```
function f = computevectorlength(v)

% function f = computevectorlength(v)
%
% <v> is a vector
%
% Return the length of vector <v> by computing
% the square-root of the sum of the squares
% of the elements of <v>.
%
% Example:
% computevectorlength([1 1])

f = sqrt(sum(v(:).^2));
```

- Functions can have multiple inputs and multiple outputs. Example of the syntax:

```
[a,b] = computesomething(x,y)
```

where `x` and `y` are two distinct inputs and `a` and `b` are two distinct outputs.

7. Function handles, anonymous functions

- The `@` symbol allows one to create a **function handle**, which is a reference to a function. Function handles are useful for passing functions to functions.
- The `@` symbol also allows one to create an **anonymous function**. An anonymous function is a function that is defined on-the-fly and does not have an .m file associated with it. Anonymous functions can make certain programming tasks easier (since one does not have to create and save .m files).

8. Debugging

- If MATLAB crashes, the last command you issued in the command window does not take effect (e.g. if the command was an assignment, the assignment does not occur).
- To help figure out why the crash happened (if it isn't already obvious from the error message), you can use MATLAB's built-in debugger.

- The debugger essentially allows you to step through your code one line at a time. After each step, you can check the workspace, check the contents of variables, and/or issue MATLAB commands, which can all be useful for figuring out what's going on.
- The GUI includes various buttons and widgets that can be helpful when debugging.

9. Debugging-related commands

`dbstop` - tell MATLAB when (i.e. in what function) to enter debugging mode
`dbstep` - in debugging mode, this tells MATLAB to execute the next line of code
`dbstep in` - like `dbstep` except that we step into function calls (so that we can debug them)
`dbcont` - in debugging mode, this tells MATLAB to execute all remaining lines of code
`dbquit` - in debugging mode, this tells MATLAB to get out of debugging mode
`dbclear` - clear debugging breakpoints set by `dbstop`