MATLAB is a very handy program for the creation, manipulation, and analysis of data. We will begin our data explorations very simply. Our very first task will be to create some artificial data to manipulate.

1. First, invoke the program from the Start button on the taskbar.

2. We will create data from a normal distribution. This is the familiar "bell curve". (We will see as the semester progresses why this particular probability distribution is so important. For now we'll just state that a good deal of actual gathered data out in the world forms a histogram which is at least roughly symmetrical and mound shaped.) MATLAB expects you to type in commands rather than select commands from a menu. We naturally must specify a few items. We will generate data from a normal distribution with mean 5 and standard deviation 3. Now MATLAB is predisposed to matrices, so consider the following command:

   xData = normrnd(5,2,2000,1)

   This creates a variable named *xData*. The command *normrnd* will generate normal data. The first two arguments specify the mean (center) and the standard deviation (spread) of the data. The next two ensure that *xData* is a matrix with 2000 rows and 1 column.

3. If you are like most people, it's hard for you to look at 200 numbers and make much sense of them. First, clear your screen with the command

   clc

   Now, in order to suppress the output to the screen, append a semicolon to your command as

   xData = normrnd(5,2,2000,1);

   If you would like to verify that fresh data has been created you may just type the variable name

   xData

   and you will again see your data. If you are oriented towards spreadsheets, you may work with your data in the variable workspace. Go to *Desktop-¿Workspace* and double click on the variable you created.

Now that we have some data, let's create a pictorial representation of the data by producing a histogram. The simplest way to do this is just with the command

hist(xData)

The figure you obtain shouldn't surprise you. The histogram should be centered at around 5 and should run from around 5-3*2=-2 to 5+3*2 = 11. However, MATLAB will, by default, produce a histogram with 10 "bins" (count your bars) placed between the minimum and maximum values in your data array. We would like a little control over how the histogram is created. To do this we need a few basic concepts.

Creating Arrays We have already (passively) created an array, *xData*. We may create an array of "bin centers" (to specify where we want our bins to be centered) for use with the command *hist* or we may creat an array of "bin edges" (to specify where we want our bins to begin and end) for use with *histc*. We'll first create bins with centers at integer values between -5 and 15. To create an array of numbers -5, -4, . . . 15 we'll type

```
binCenters = -5:1:15
```

You may use a semicolon to suppress output if you'd like. The generic form of the assignment is

```
variableName = startValue:stepSize:stopValue
```

For instance, to create the array [6 11 16 21], you'd type

```
smallArray = 6:5:21
```

What does

```
smallArray = 6:5:23
```

produce?

We're now ready to create our new histogram. Enter the command

```
hist(xData,binCenters)
```

How many bars do you have now?

Suppose instead you'd like to specify where your bins start and stop. Create bins with edges separated by a distance of 0.5, starting at -5 and ending at 15 and store your result in a vector named *binEdges*. Then, create a histogram with the command

```
histc(xData,binEdges)
```

Where's our picture?!?

If you read the help information on *histc*

```
help histc
```

you'll notice that *histc* does your counting for you, but doesn't produce a plot. (You should also note how that last bin is created). In order to "see" our data do the following:

```
histData = histc(xData,binEdges)
bar(binEdges,histData)
```

The only problem now is that each bar is *centered* at the integer values -5, -4.5, . . . , 15, whereas we'd like them centered on -4.75, -1.25, . . . , 14.75. You might want to "zoom in" on your graph to verify this. One (awkward) way to fix the centering issue is with the commands

```
barCenters = -4.75:0.5:15.25;
bar(barCenters,histData)
```

The last bar should be of height 0. Finally, to make your bars have unit width, type the command

```
bar(barCenters,histData,1)
```

Simple Plotting We'll spend a good deal of time computing theoretical distributions and plotting data to confirm our work. To construct a very simple plot consider the following commands.

```
x = 1:2:15
y = 2*x + 1
plot(x,y)
```

Or, for more fun

```
plot(x,y,'r*')
```

Note that the single quotes are important. Our simple plot will be that of the probability density function of a normal distribution. Generate points as follows.

```
x = -20:0.1:20;
y = normpdf(x,4,3);
plot(x,y,'bo')
```

After we have discussed modified relative frequency histograms we can bring the ideas of plotting a curve and plotting "data" together to experimentally verify all of the theory we derive in class.

Finally, if time permits, have a little fun by generating uniformly distributed data on the interval $[0, 1]$ twice, then looking at a histogram of the sum with the commands

```
x1 = rand(1000,1);
x2 = rand(1000,1);
s = x1 + x2;
hist(s)
```

Does the histogram look like you thought it would?