

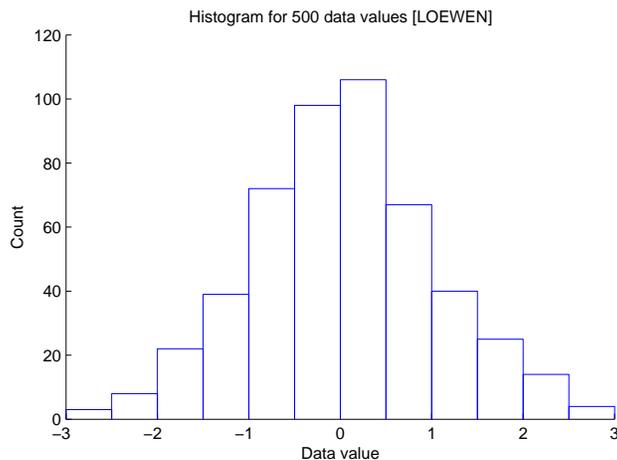
MECH 221 Computer Lab 1: Histogram Lab Activities

OVERVIEW: *Producing a Histogram*

The main product of this lab is a Matlab function named `hyst` that works like this:

```
data = randn(1,500);           % Build a 1-by-500 matrix with random entries, mean 0
nodes = linspace(-2,2,21);    % Set up the nodes between histogram bins
counts = hyst(data,nodes)     % Count data values in each bin, and plot the results
```

The function `hyst` will return a vector of counts corresponding to the specified data bins, print a warning if there were any outliers that didn't get binned, and generate a plot similar to this one:



The name of the function must be `hyst`. This allows for automated testing. Also, the more obvious name `hist` is already in use for a built-in Matlab function with a similar purpose; the Matlab built-in named `histc` is even closer to `hyst`.

Students will also submit proof that their function works correctly. Detailed instructions concerning what must be handed in appears at the end of this document.

SPECIFICATIONS: *Inputs, outputs, and actions for function hyst*

Function `hyst` has 2 input arguments, `data` and `nodes`, in this order. It has a single return value, `counts`.

`data` is a row or column vector containing real numbers. The number of elements in `data` is limited only by the capacity of the host computer.

`nodes` will be a row or column vector with 2 or more elements, presented in increasing order. Function `hyst` will print an error message and return the empty matrix if the given input does not satisfy these two assumptions.

Let N denote the number of elements in `nodes`, and use the mathematical notation x_k for the Matlab value `nodes(k)`. The long interval $[x_1, x_N)$ determines the range of data to be analyzed. The intermediate values divide the full interval into $N - 1$ adjacent subintervals, namely,

$$I_1 = [x_1, x_2), I_2 = [x_2, x_3), \dots, I_{N-1} = [x_{N-1}, x_N).$$

`counts` will be a row vector with $N - 1$ elements, evaluated and returned by `hyst`. The vector entry `counts(k)` will be the number of given data values lying in the interval I_k defined above.

Numbers in the vector `data` that lie outside the long interval $[x_1, x_N)$ are called *outliers*. If there are any outliers, function `hyst` will print a warning message that includes the number of outliers detected.

Function `hyst` will also produce a plot like the one shown above. The plot title must include the number of data points and the family name of the student submitting the work. The plot itself will show $N - 1$ rectangles aligned with the coordinate axes: for each k , rectangle number k will capture the (x, y) points satisfying

$$x_k \leq x < x_{k+1}, \quad 0 \leq y \leq \text{counts}(k).$$

CONSTRUCTION HINTS

The steps suggested below will guide you through the production, testing, and delivery of the function `hyst`. They break the work of building the function in tiny steps, making sure each new feature works correctly before taking the next step. Although the steps are suggested, not required, experience shows that incremental development with plenty of testing is an efficient and effective approach.

Each step describes new code to be added to a growing function file. But these additions cannot always be simply typed onto the bottom of the file from the previous step: new material must sometimes be inserted above earlier content, and earlier typing may need to be removed or changed to complete some steps.

- Create a new file named `hyst.m`, in which the first line is an exact copy of this function header:

```
function counts = hyst(data,nodes)
```

Add a comment line showing your UBC Student ID number and name. Then add just enough active Matlab code to complete the definition of a function that does absolutely nothing, but returns the empty matrix, `[]`. Check that this works: entering `hyst(1,2)` at the Matlab prompt should give a return value of `[]`.

- Add some input checking: insert lines that detect whether the input vector `nodes` has 2 or more elements. If not, display an error message and return the empty matrix. If so, print nothing but return a row vector with $N - 1$ components, each with the value 0. (Here, as in the specifications above, N is the number of elements in the vector `nodes`.) Test.
- Do more input checking: compare the given vector `nodes` with the vector `sort(nodes)`. If these are not identical, display an error message and return the empty matrix. If they are, proceed as before to return a 0-vector of dimension $N - 1$. Test.
- Count the data. Two obvious approaches present themselves; choose one (ideally for sensible reasons) and stick to it:

- (a) Use a `for`-loop that visits each element of the `data` vector, like this:

```
for x=data
    ...
end;
```

In each pass through the loop, the variable `x` will hold one of the data values. You decide which interval I_k , if any, contains the value x , and increment the associated counter value in the vector element `counts(k)`.

(b) Use a `for`-loop that visits each subinterval in order, like this:

```
for k=1:(N-1)
    ...
end;
```

In each pass through the loop, the variable `k` will hold an integer you can use to define the interval I_k . Look through the data vector to see how many elements belong in that interval, and store the count you get in the vector element `counts(k)`.

In approach (a), you may need to use an embedded `for`-loop over `k` in `1:(N-1)` to identify the correct interval; in approach (b), you may need to embed a `for`-loop over `x` in `data` to identify the elements to count. (Efficient alternatives exist and are encouraged, but not required.)

- Test vigorously. Start with a small number of nodes (say, 3) and just a few data values. Entering the command

```
data = [3,3,3,3,3,1,2,2]; nodes = [0,1.5,2.5,10]; hyst(data,nodes)
```

should produce the answer `[1,2,5]`. Experiment with outliers. Make sure data values that land exactly on a node boundary get counted once and only once.

- Add outlier detection. Count the number of elements in the vector `data` that do not fit into any of the intervals I_k , $1 \leq k \leq N - 1$. If that number is not 0, print it, along with a warning message. Test.
- Set up for plotting. Use a suitable modification of this command to get a quick and dirty approximation to the bar-chart you eventually want:

```
plot(nodes(1:(N-1)),counts);
```

Figure out how to add a title including your family name, and suitable axis labels. Test.

- Beautify the plot. Replace the `plot` line from the previous step with a `for`-loop that visits each subinterval I_k and draws a line joining the following (x, y) -pairs, in order:

$$(x_k, 0), (x_k, c_k), (x_{k+1}, c_k), (x_{k+1}, 0), \quad \text{where } c_k = \text{count}(k).$$

Remember to say `hold on` at an appropriate point. Test.

- Test your function more aggressively, perhaps copying the 3 lines used for illustration in the Overview section above. Repeat several times: sometimes the random number generator produces outliers and sometimes it doesn't. Then look ahead to the submission requirements and make sure you get reasonable results for the data and nodes provided there.

SUBMITTING YOUR WORK

Hand in the 3 items detailed below. Ideally these will fit on 3 sheets of paper.

1. A printed copy of your file that defines the function `hyst`. Make sure that the file includes your name and UBC Student ID, so that these critical items appear in the printed material (not just in handwritten form that you may choose to add later).
2. A printed copy of the program output (text, if any, plus plot) for the small daily test set. Get the test set by downloading from Connect the file on this list that is named for your assigned lab day:

```
smallMONTtest.mat, smallTUEtest.mat, smallWEDtest.mat, smallFRITest.mat.
```

Giving the command `load('smallDDDtest.mat')`; at the command prompt should create new vectors named `DDDdata` and `DDDbnodes`. (Expect `DDD` to be replaced with your 3-letter weekday code.) Apply `hyst`, print the output, and hand it in. Note that your family name must appear in the title of the plotted histogram as printed by the computer—not just in whatever handwritten decorations you may choose to add later.

3. A printed copy of the program output (text, if any, plus plot) for the large daily test set. To get this, follow the instructions for item 2, but use the file name `bigDDDtest.mat` to create vectors `DDDbdata` and `DDDbnodes`.

HINTS

Your professor's version of `hyst` uses the following built-in Matlab functions and operations. Matlab's built-in help system provides details on each of these.

<code>.*</code>	elementwise multiplication of matrices (inline form of function <code>times</code>)
<code>[]</code>	the empty matrix
<code>disp</code>	display array
<code>figure</code>	create figure window
<code>for ... end</code>	repeat statements a specific number of times
<code>function</code>	add new function
<code>hold on</code>	hold current graph
<code>if ... end</code>	conditionally execute statements
<code>length</code>	length of vector (i.e., number of components)
<code>norm</code>	matrix or vector norm (i.e., magnitude of vector quantity)
<code>plot</code>	linear plot
<code>return</code>	return to invoking function
<code>sprintf</code>	write formatted data to string
<code>sum</code>	sum of elements
<code>title</code>	graph title
<code>xlabel</code>	<i>x</i> -axis label
<code>ylabel</code>	<i>y</i> -axis label
<code>zeros</code>	zeros array