ENGR 101 – Chatper 7

Statistics and Simulation

Laura Alford, James Juett, Rick Niciejewski 9/19/2020

Review: `min/max`

- The `min` and `max` functions can be used to find the smallest or largest elements in a vector. They too, work in each column first and must be applied twice for a whole matrix.
- `min` and `max` have a compound return value. They return both the value found, and also the index where it was found.

```
[m, i] = max(X)
```

Review: The `sort` Function

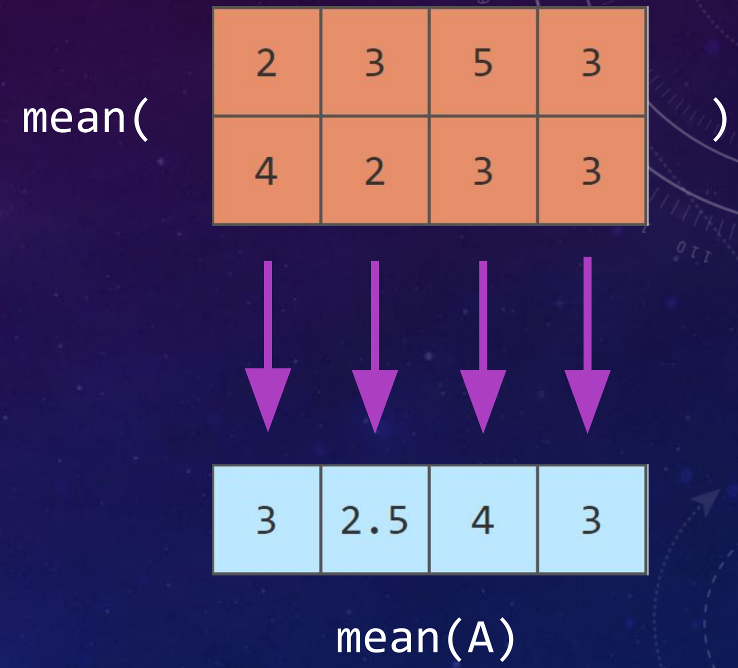
- By default, the `sort` function works with column vectors. (If you have a matrix, each column is sorted individually.)
- `sort` uses a compound return to give us both a sorted version of the vector AND a vector of sorted indices.
- You can provide 'ascend' or 'descend' as a another argument to specify order.

```
[S, I] = sort(X, 'ascend')
```

	X	S	I
1	8	1	6
2	2	2	2
3	9	4	5
4	5	5	4
5	4	8	1
6	1	9	3

mean

- The mean function returns the column-by-column mean of a matrix.
  (Or for a single row, the mean of that row.)



2	3	5	3
4	2	3	3

```
A = [2,3,5,3;4,2,3,3]
```

Row-by-Row

- An extra argument to functions like sum, mean, sort, etc. allows you to specify the dimension to operate on.

`A = [2,3,5,3;4,2,3,3]`

`mean(A,2)`

Specifying the 2^{nd} dimension allows you to do a row-by-row mean instead.

Applying to all elements

- The `mean` function returns the column-by-column mean of a matrix. (Or for a single row, the mean of that row.)

2	3	5	3
4	2	3	3

`A = [2,3,5,3;4,2,3,3]`

3	2.5	4	3

`mean(A)`

3.125

`mean(mean(A))`

3.125

`mean(A(:))`

3.125

`mean(A, 'all')`

median

- The `median` function computes the median of a dataset.
- It works with arrays in the same way as the mean function.
  - i.e. column-by-column, selecting dimensions, etc.

- The median of a dataset is the number that would appear in the middle if the numbers were sorted.
  - In case of an even number of elements, average the two in the middle.

B	5	0	8	9	6	7	7	3	5	1
sort(B)	0	1	3	5	5	6	7	7	8	9
median(B)	5.5									

mode

- The mode of a dataset is the value that occurs most often.
- The `mode` function returns this value.

X	1	9	0	7	8	8	0	3	7	8

mode(X)	8

- Using a compound return, you can also get the frequency.

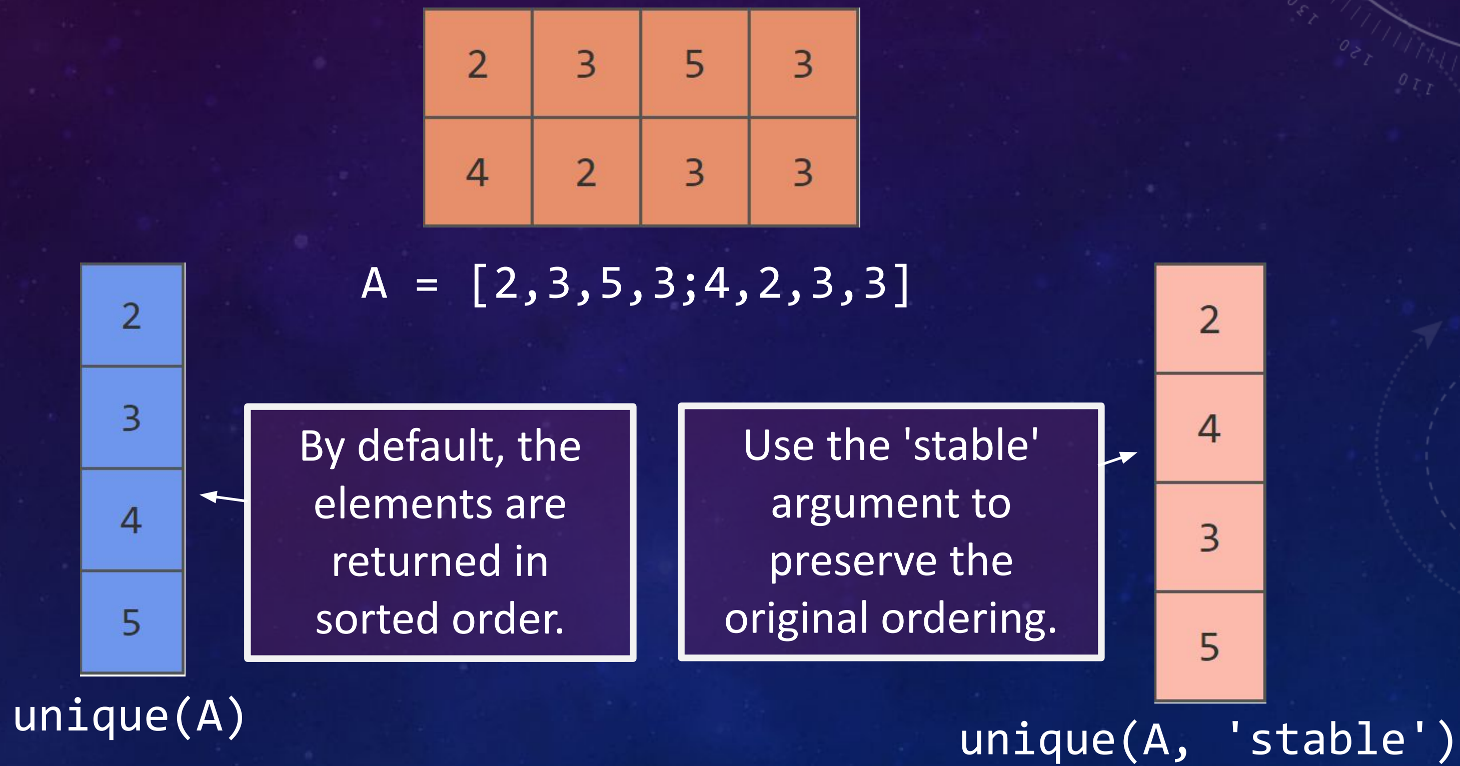
X	1	9	0	7	8	8	0	3	7	8

```
[m, freq] = mode(X)
```

m	8	freq	3

unique

- The `unique` function takes an array as input and returns a vector of its unique elements (i.e. with duplicates removed).



3 min

Exercise: City Latitude Statistics

- Open the `CityLatitudes.m` file from the google drive.

```
% ...Code above to read in data and store names, populations, latitudes, longitudes

[maxLat, iMaxLat] =                 % complete to find the max latitude
[minLat, iMinLat] =                 % complete to find the min latitude
meanLat =                           % complete to find the mean latitude
medianLat =                         % complete to find the median latitude
[modeLat freqModeLat] =             % complete to find the mode of the latitudes


% Display a summary of the latitude statistics
disp(['The most northern city is ' names{iMaxLat} ' (' num2str(maxLat) ' deg)']);
disp(['The most southern city is ' names{iMinLat} ' (' num2str(minLat) ' deg)']);
disp(['The mean latitude is ' num2str(meanLat) ' deg']);
disp(['The median latitude is ' num2str(medianLat) ' deg']);
disp(['The mode latitude is ' num2str(modeLat) ' deg']);
```

`num2str()` converts a numerical value to its string equivalent

concatenate strings by creating a vector using [ ]

Solution: City Latitude Statistics

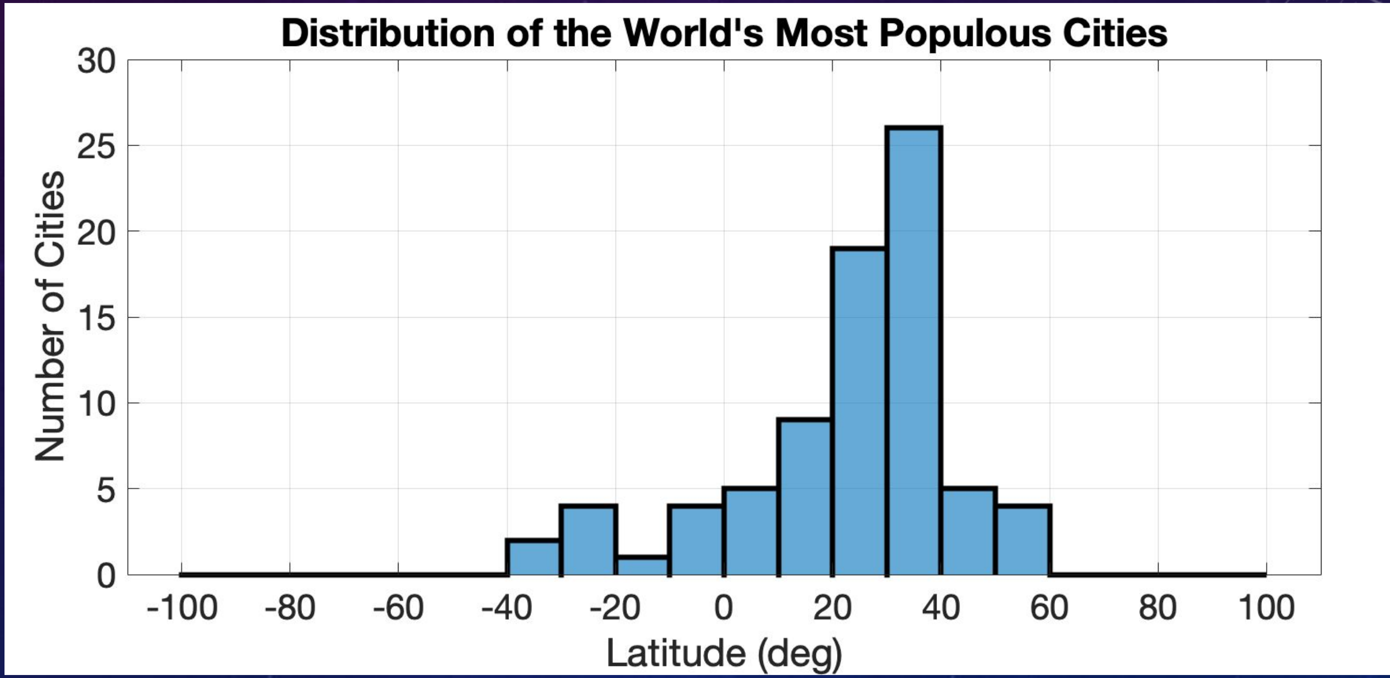
```
...
% Code above to read in data and store ...
[maxLat, iMaxLat] = max(lat);
[minLat, iMinLat] = min(lat);
meanLat = mean(lat);
medianLat = median(lat);
[modeLat freqModeLat] = mode(lat);

% Display a summary of the latitude statistics
...
```

```
>> CityLatitudes
The most northern city is Saint Petersburg (59.95 deg)
The most southern city is Cape Town (-33.9333 deg)
The mean latitude is 22.1584 deg
The median latitude is 26.9333 deg
The mode latitude is 23.0333 deg
```

Histograms

- In general, a histogram is a visualization of the frequency of occurrence for certain values in a dataset.



A spectacular guide to histograms here: http://tinlizzie.org/histograms/

Creating a Histogram

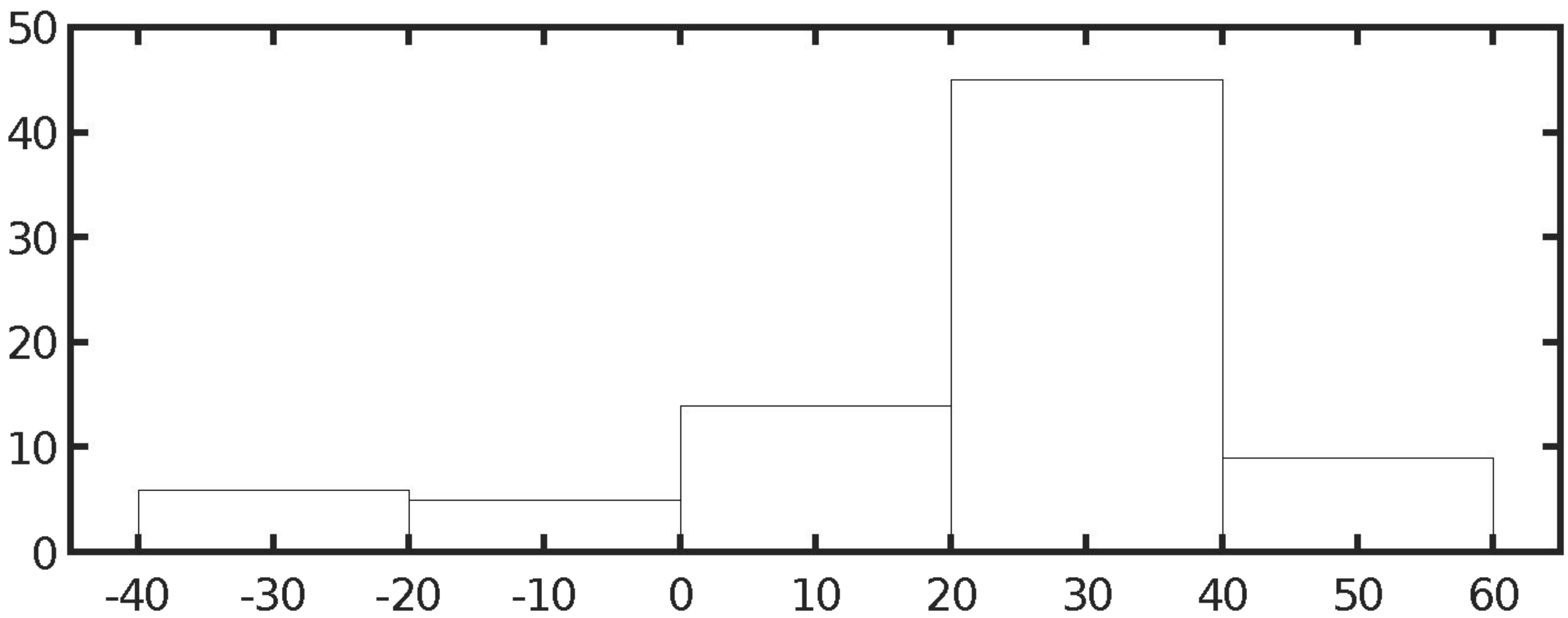
- Use the `histogram` function

lat

```
31.2000
24.8667
39.9000
28.6167
6.45000
39.1333
41.0167
35.6833
23.1333
18.9833
⋮
```

vector of data points

```
histogram(lat);
```



Note: There's also a `hist` function that does some of the same things, but `histogram` is generally better.

Histogram Bins

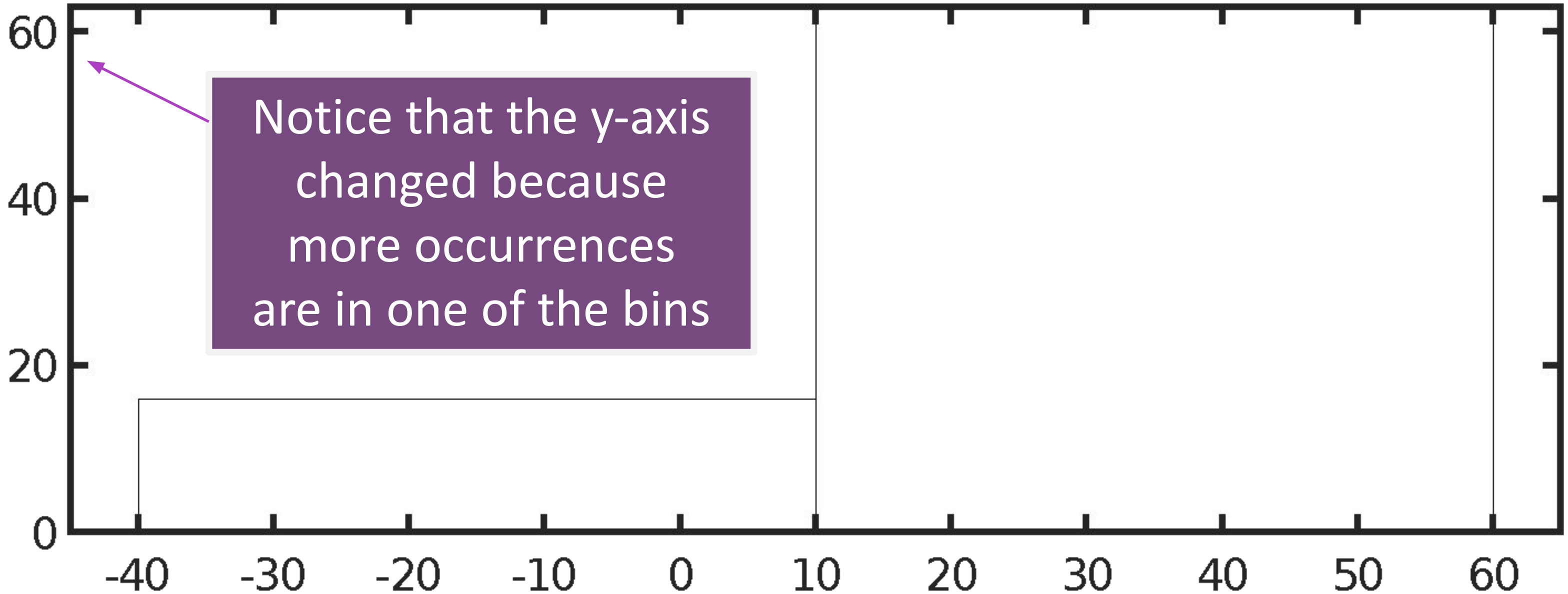
- You can specify the number of "bins" you want to use:

```
histogram(lat, 2);
```

number of bins

lat

```
31.2000
24.8667
39.9000
28.6167
6.45000
39.1333
41.0167
35.6833
23.1333
18.9833
⋮
```



Note: There's also a `hist` function that does some of the same things, but `histogram` is generally better.

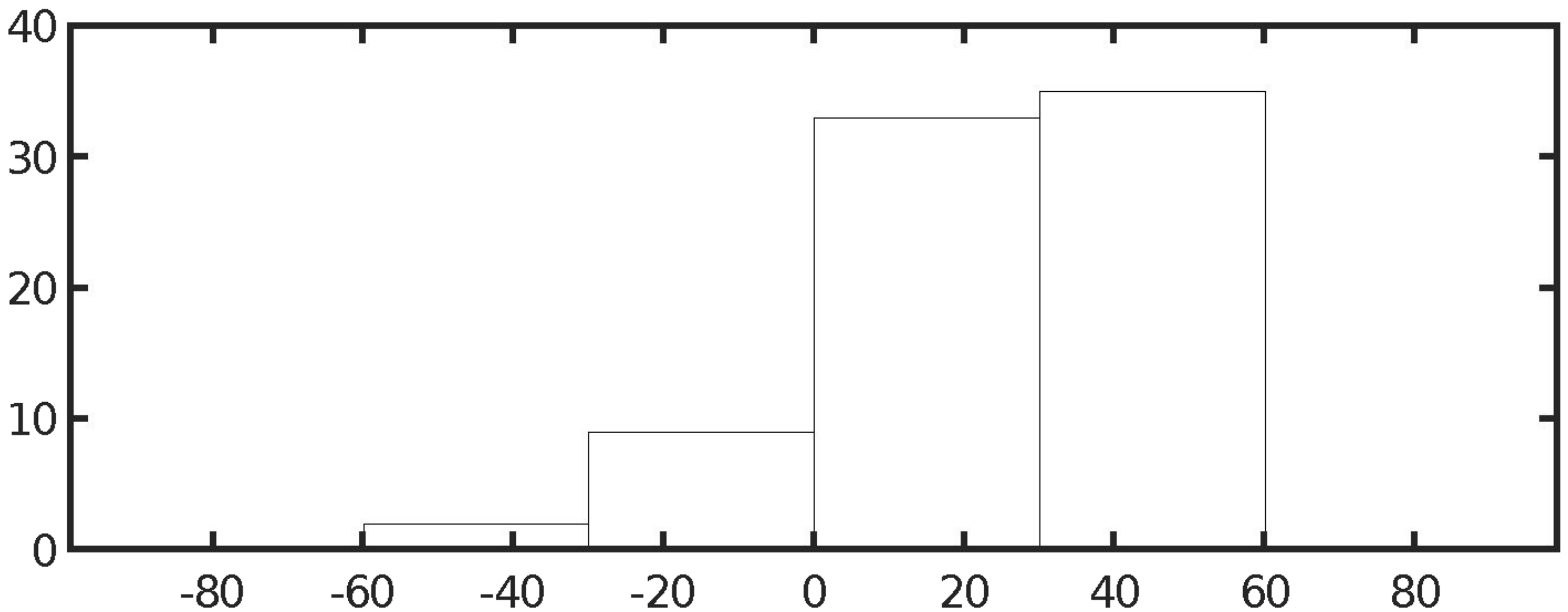
Customizing Histogram Bins

- MATLAB will try to pick reasonable "bins" for you, but you can also specify the bounds explicitly.

```
histogram(lat, -90:30:90);
```

same as
-90, -60, -30,- 0, 30, 60, 90

```
lat
31.2000
24.8667
39.9000
28.6167
6.45000
39.1333
41.0167
35.6833
23.1333
18.9833
⋮
```



Note: There's also a `hist` function that does some of the same things, but `histogram` is generally better.

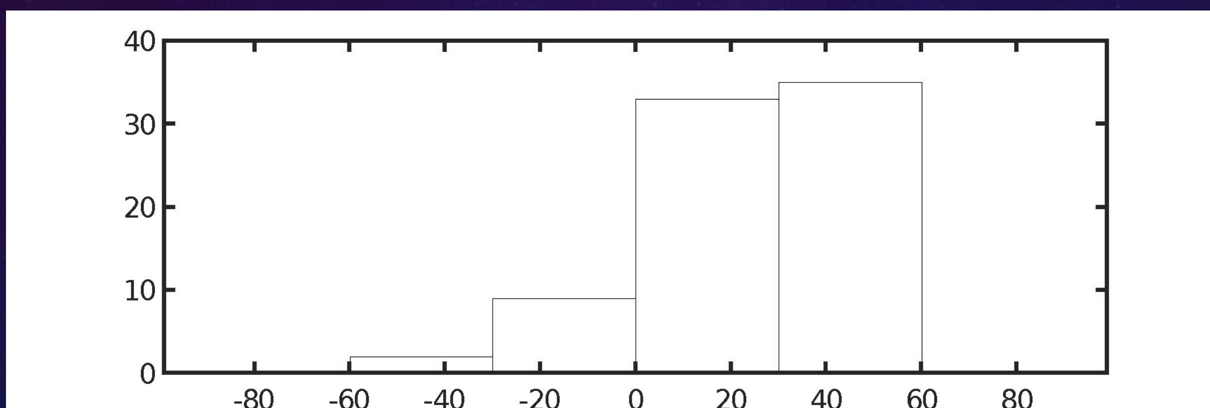
histcounts

lat

```
31.2000
24.8667
39.9000
28.6167
6.45000
39.1333
41.0167
35.6833
23.1333
18.9833
⋮
```

- The `histcounts` function gives you the number of elements belonging to each histogram bin.

```
histogram(lat, -90:30:90);
```



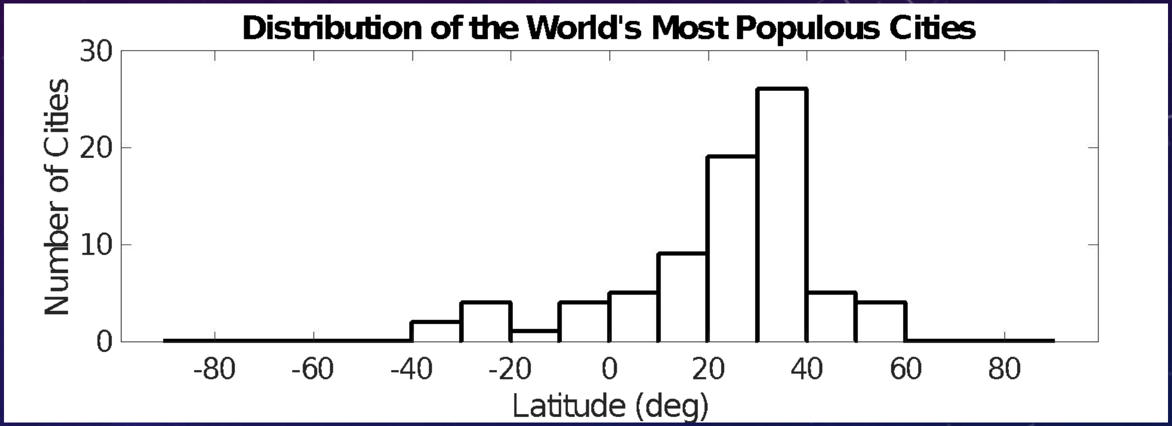

```
histcounts(lat, -90:30:90);
```

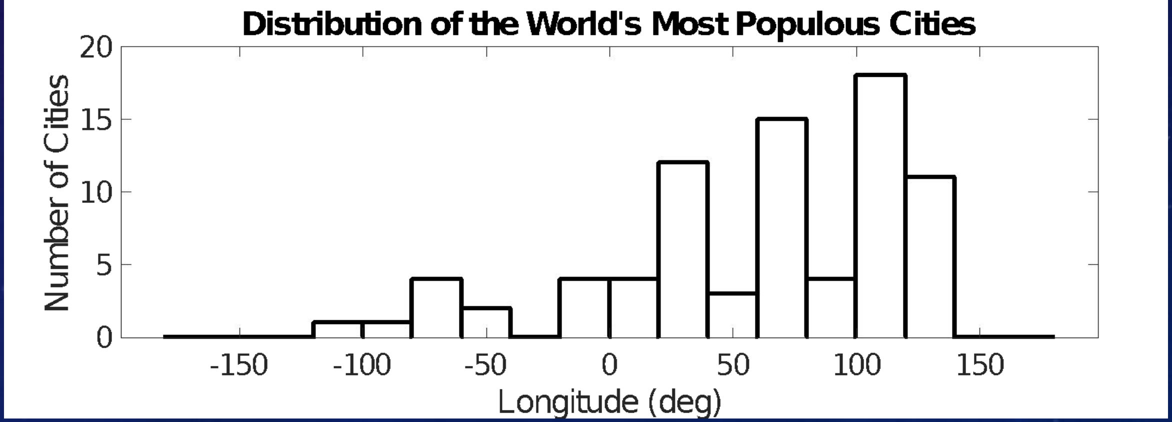
0	2	9	33	35	0

Exercise: City Longitude Histogram

- Start with the `MakeCityLatitudeHistogram.m` script in the google drive.
- Copy and paste the code that makes the <u>latitudes</u> histogram.
- Make appropriate changes to the copied code to generate a histogram of the <u>longitudes</u> of the cities.

3 min





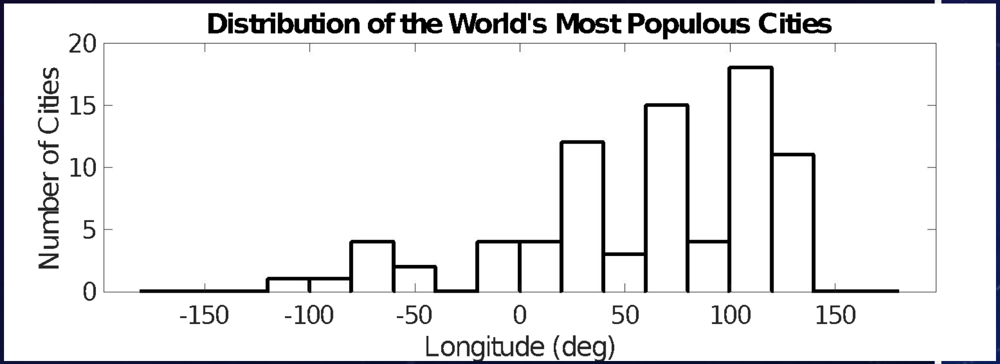
Solution: City Longitude Histogram

```
%% Make histogram of cities' longitude

fig = figure();
bins = -180:20:180;
h = histogram(lon,bins);

% change the defaults so the plot is better for a presentation
h.LineWidth = 3;
ax = gca;
ax.FontSize = 20;
grid on;

% add some labels
xlabel('Longitude (deg)');
ylabel('Number of Cities');
title('Distribution of the World''s Most Populous Cities');
```

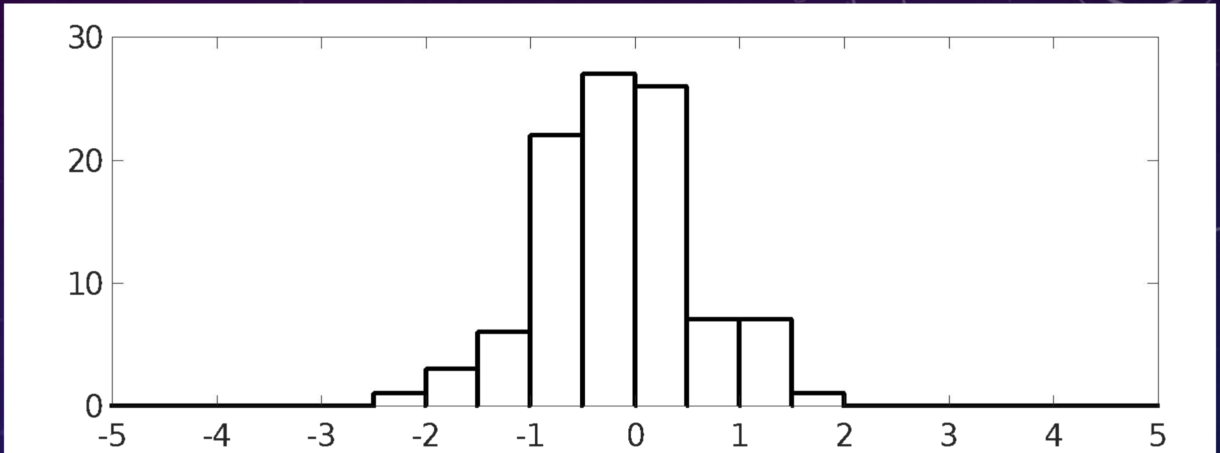


Variance and Standard Deviation

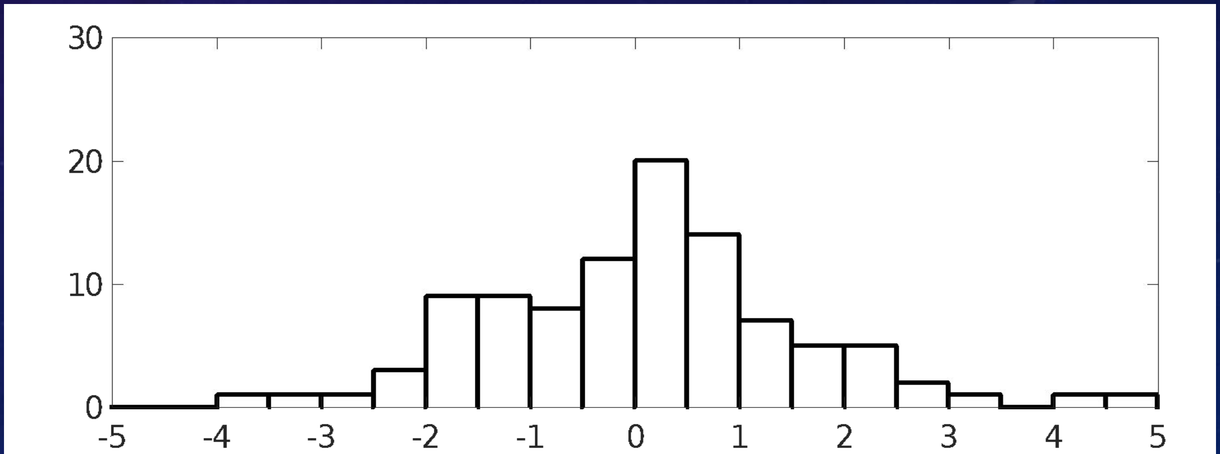
- Variance and standard deviation are measure in descriptive statistics that tell us how spread out a dataset is.
- MATLAB has functions to calculate these:
  - `var` – variance
  - `std` – standard deviation
- Both of these functions work with arrays in the same way as the mean function.
  - i.e. column-by-column, selecting dimensions, etc.

Variance and Standard Deviation

<u>Narrow Distribution</u>
var = 0.5537
std = 0.7440



<u>Wide Distribution</u>
var = 2.1975
std = 1.4824

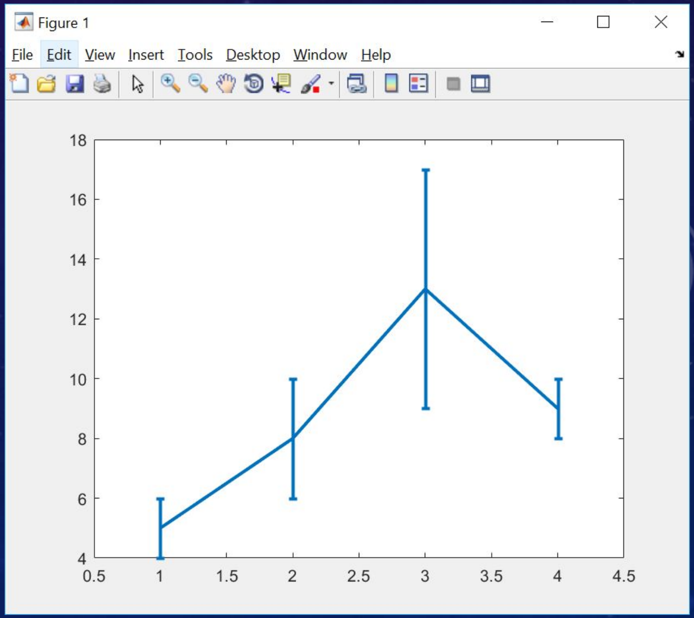


Plots with Error Bars

- Use the errorbar function to show a plot with "error bars" at each data point.
- Error bars can be used to convey a range of values for each point on the plot, or uncertainty about a measured value.
- Example:

```
x = 1:4;
y = [5,8,13,9];
err = [1,2,4,1];
h = errorbar(x, y, err);

% adjust for projector :)
set(h, 'LineWidth', 2);
```



Example: Battery Lifecycle Analysis

- Example: A company that produces smartphones wants to analyze battery lifetime throughout several years of use.
  - A set of batteries have been put through several years of simulated use.
  - Twice per "year", the lifetimes of each battery were tested and the mean and standard deviation for the set was recorded.
  - We would like to visualize the degradation of battery lifetime throughout the years as well as the amount of variability.
    - e.g. Can we claim that the phone had 3 hours battery life when new?
    - e.g. Can we claim after 2 years, the phone will still have 2 hours of battery life?

To follow along:

```
load('batteryLife.mat');
```

Solution: Battery Lifecycle Analysis

```
% create a plot with error bars
h = errorbar(time, batteryMean, batteryStdDev);

% use the graphics object to set the line width
set(h, 'LineWidth', 3); % wider than usual for projector

% Add title and axis labels
title('Battery Life Over Time');
xlabel('Years of Use');
ylabel('Full-charge Battery Life (hours)');

% use gca to set properties
ax = gca;
ax.FontSize = 20;
ax.XLim = [0,5];
ax.YLim = [0,3.5];
ax.YGrid = 'on';
```

Break Time

We'll start again in 5 minutes.

Analyzing Existing Data vs. Simulation

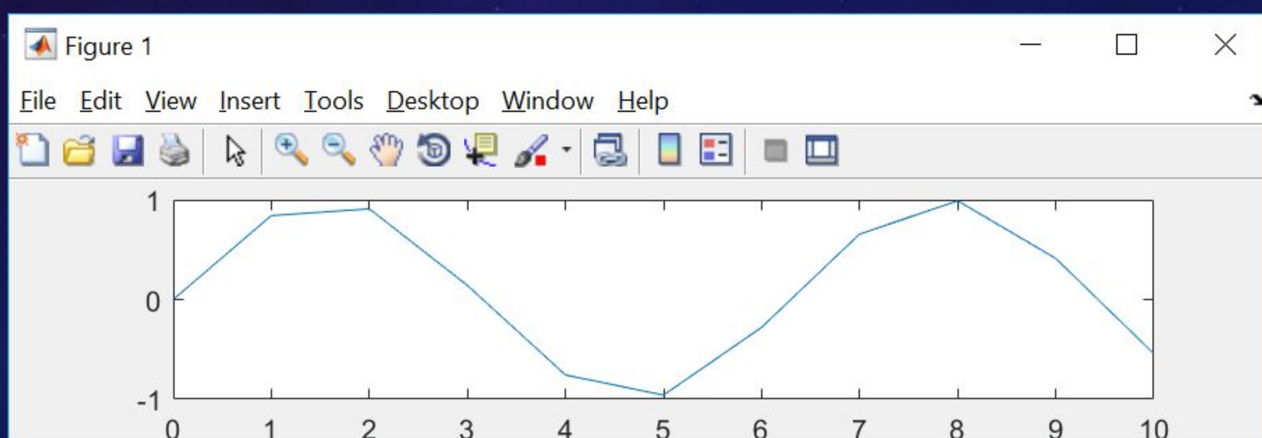
- The examples we've been working with have mostly been about analyzing existing data.
  - Read data in from a file
  - Look for some things/process the data
  - Plot the data to see trends, patterns, extremes, etc.
- MATLAB is also widely used to simulate data
  - Math equations are converted to computer programs
  - Predicted data is generated by the program based on a starter set of data
  - Look for some things/ process data/ plot data to see trends, etc.

Plotting Functions

- Can we use MATLAB like a graphing calculator?
  - This doesn't work: `plot(sin);`
- `plot` requires a set of ordered pairs
  - We can generate these easily using vectorized code!

```
x = 1:10;
y = sin(x);
plot(x,y);
```

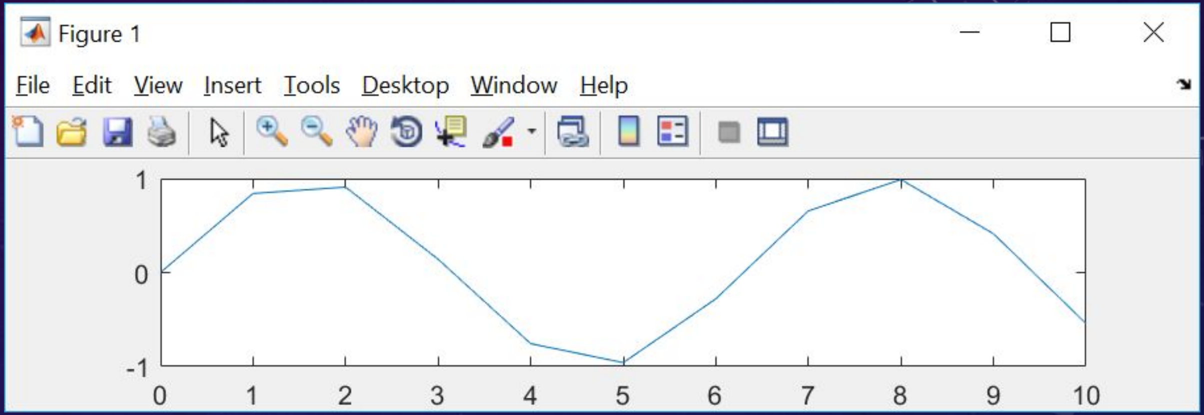
x	0	1	2	3	4	5	6	7	8	9	10
y	0	0.84	0.91	0.14	-0.76	-0.96	-0.28	0.66	0.99	0.41	-0.54



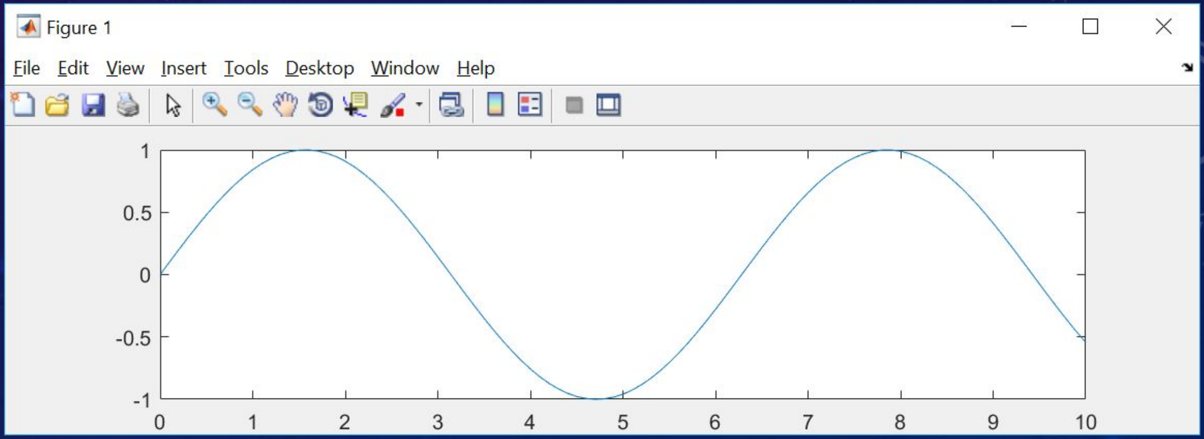
Plotting Functions

- To create a smoother plot of a math function, just use more data points. To do this with a range, decrease the step size.

```
x = 0:10;
y = sin(x);
plot(x,y);
```




```
x = 0:0.1:10;
y = sin(x);
plot(x,y);
```



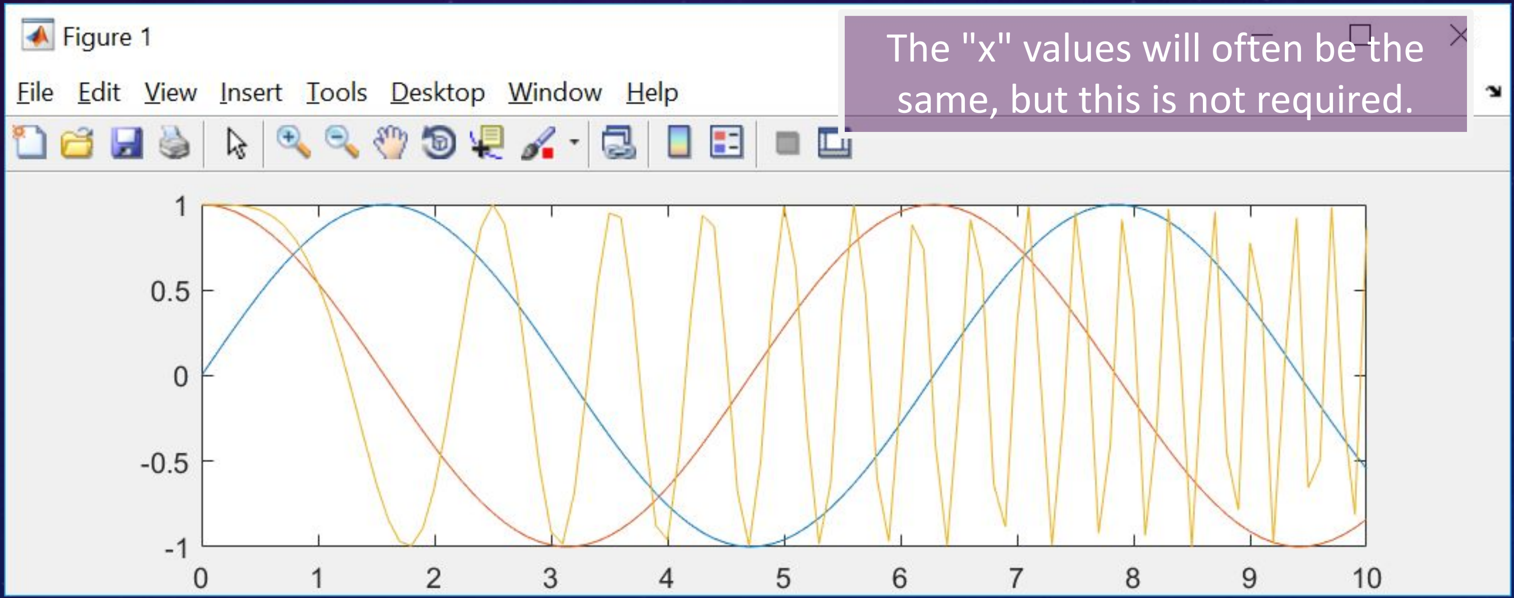
Recall: Plotting Multiple Sets of Data

- `plot` allows you to specify multiple sets of data to be shown together.

Change to `x = 0:0.01:10;`

```
x = 0:0.1:10;
plot(x, sin(x), x, cos(x), x, cos(x.^2));
```

Why does the plot of *cos(x^2)* look so bad?



The "x" values will often be the same, but this is not required.

The `linspace` Function

- The `linspace` function provides an alternate way to create evenly spaced vectors of numbers.
- Range Notation:

```
0:0.01:10
```

Start

Step size

End

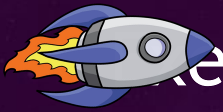
- `linspace`:

```
linspace(0,10,1000)
```

Start

End

How many numbers?



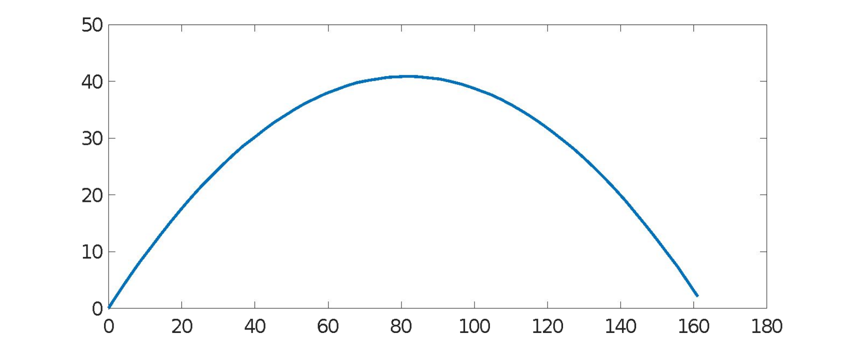
Exercise: Projectile Motion

- Assume a baseball is thrown with:
  - `v0` – initial speed
  - `theta` – initial angle

- Then its position over time is given by:
  - `x = v0 .* cos(theta) .* t`
  - `y = v0 .* sin(theta) .* t - 9.8 .* t.^2 ./ 2`

- Plot its (x,y) position for an initial speed of 40m/s and angle of pi/4 radians (45 degrees) over the course of 15 seconds.
  - Hint: Create a "starter set" of data for `t` consisting of equally spaced numbers using either `linspace` or the range operator `(:)`. Then calculate x and y.
  - Bonus: Plot only the portions of the graph for which the projectile would be above the ground (i.e. y > 0).

Solution: Projectile Motion




```
% set initial conditions
v0 = 40; % initial speed in m/s
theta = pi/4; % initial angle in radians

% create starter data
t = 0:0.1:15; % time in seconds

% simulate horizontal and vertical motion
x = v0 .* cos(theta) .* t;
y = v0 .* sin(theta) .* t - 9.8 .* t.^2 ./ 2;

% plot vertical vs. horizontal motion
plot(x(y>=0),y(y>=0),'LineWidth',3);
grid on;
ax = gca;
ax.FontSize = 20;
```

Sampling From Random Distributions

- MATLAB provides functions for sampling from a variety of probability distributions.
- This allows us to run simulations of random phenomena if we know the parameters of their distributions.
- For example, to simulate a uniform discrete distribution:

3	1	6	6	3

`randi(6,1,5)`

Create a 1x5 matrix sampled from a uniform discrete random distribution.

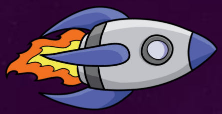
10	7	19	8	3

`randi(20,1,5)`

Simulate rolling a 20-sided die 5 times.

Probability is Hard. Vectorization is Easy.

- Question: If I roll a 6-sided die and a 20-sided die, what is the probability that the sum is greater than 15?
- Math Answer:
  - There's certainly an answer, but this isn't a probability class.
  - Sometimes (e.g. the election example from lecture 6 where we're simulating over 200 different countries), there *is* no clean math answer.
- Computational Answer:
  - Simulate this a bunch of times with MATLAB and see how often we get a sum that's greater than 15.
  - Because this involves "repetition", vectorization is our tool of choice.



5 min

Exercise: Computational Random Simulation

- Question: If I roll a 6-sided die and a 20-sided die, what is the probability that the sum is greater than 15?
- Take these steps to find the answer:

1. Generate row vectors for 10000 rolls of each kind of die using the `randi` function from the previous slide.
2. Add the row vectors together to get the sum.
3. Use logical indexing to select sums greater than 15 and count them.
4. Divide by 10000.
5. Profit.

Solution: Computational Random Simulation

- Question: If I roll a 6-sided die and a 20-sided die, what is the probability that the sum is greater than 15?

```
numTrials = 10000;

% roll the dice
rolls6 = randi(6, 1, numTrials);
rolls20 = randi(20, 1, numTrials);

% compute the sum for each roll
rollsSum = rolls6 + rolls20;

% count how many are > 15
numSuccess = sum(rollsSum > 15);

% compute probability
prob = numSuccess / numTrials;
```

Recall: Election Forecasting

- The `means` and `stddevs` variables provided with the election workspace are column vectors containing the parameters of the distributions for each state.

states	means	+	stddevs	.*	randn(size(states))	⇒	
Afghanistan	67.2433		3.5706		-0.1771		70.9095
Albania	52.1349		9.634		-0.3001		63.0522
Algeria	34.9601		2.4889		1.0005		37.59
American Samoa	50.9471		6.1881		-0.6995		59.0384
Andorra	47.5475		3.0923		0.0599		51.2639
Angola	47.0138		3.35		0.5761		50.2289
Anguilla	55.9793		4.7602		0.0761		60.5013
Antigua and Barbuda	53.3673		4.223		-0.2184		57.5254
Argentina	46.4339		2.481		1.4155		47.9186
Armenia	54.4079		4.1308		1.4881		57.1157
⋮	⋮		⋮		⋮		⋮

Other Distributions

- MATLAB supports many different random distributions.

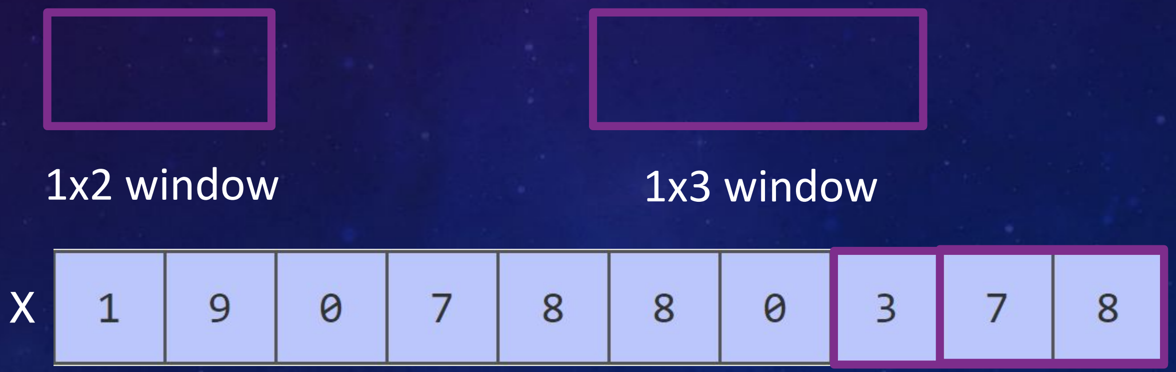
- Discrete:
  - Binomial
  - Multinomial
  - Geometric
  - Poisson
  - etc…

- Continuous:
  - Beta
  - Gamma
  - Exponential
  - Chi-square
  - etc...

- As always, consult the documentation for more details!

Convolution

- Convolution is a mathematical operation applied to a vector or matrix that computes values based on small "neighborhoods" of elements.
- We'll only consider vectors for now.
- The neighborhoods we consider are determined by a "sliding window" that moves across a vector:



Convolution

- A particular window is defined by a vector.
- The convolution overlays the window on the larger matrix:
  - First, it multiplies overlapping elements.
  - Then, it adds those results together.

1	2

1x2 window

```
conv(X,[2,1],'valid')
```

X	1	9	0	7	8	8	0	3	7	8

23	**24**	**8**	**6**	**17**	**23**

Moving Averages

- We can use convolution to compute moving averages.
  - These are useful for removing noise from measurements of a signal over time.
- Create a window vector with values that add to 1.

0.5	0.5

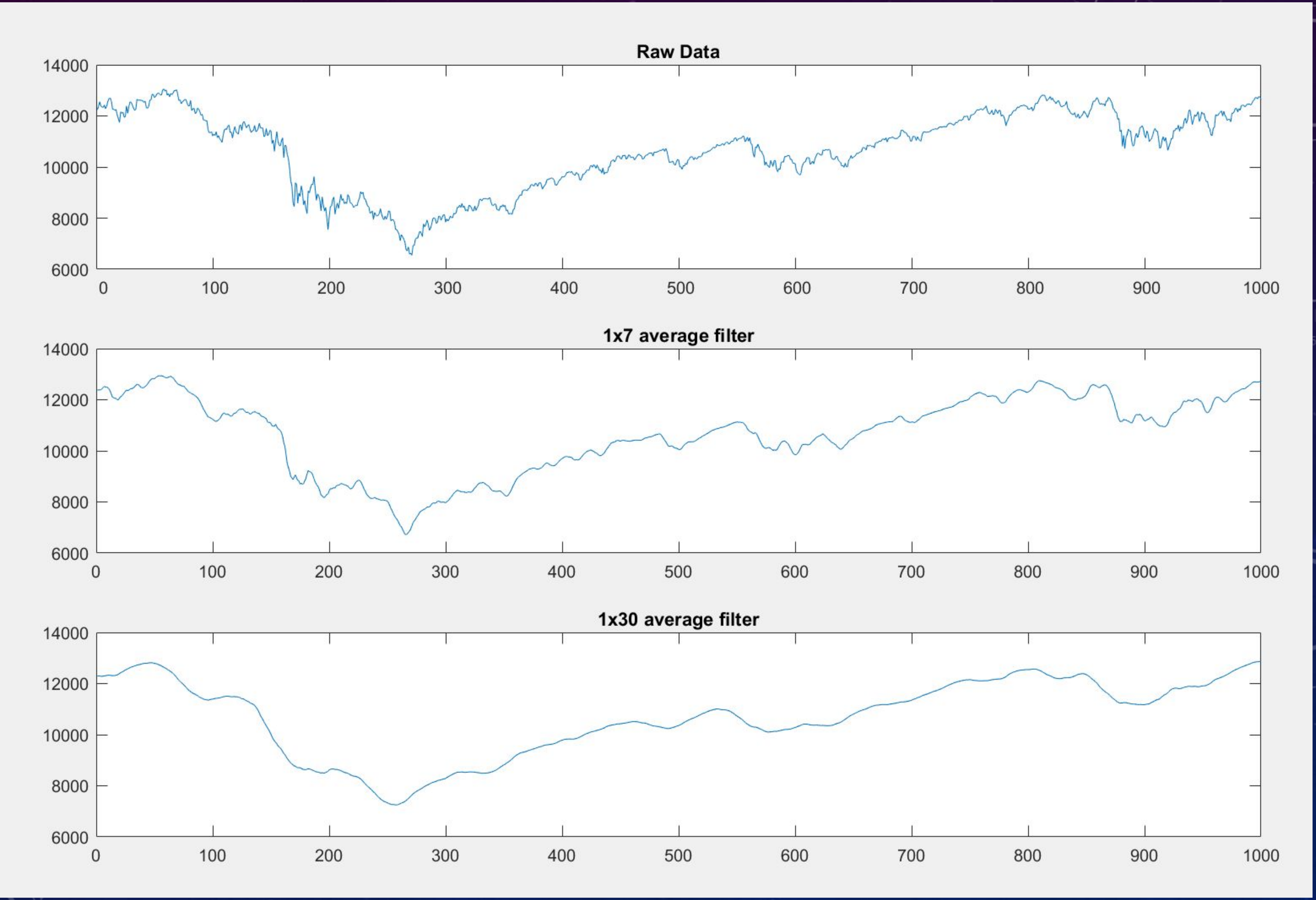
1x2 window

```
conv(X,ones(1,2)./2,'valid')
```

X									
1	9	0	7	8	8	0	3	7	8

7.5 8 4 1.5 5 7.5

Example: Dow Jones Industrial Average



Obtaining Frequency Counts

X										
X	1	9	0	7	8	8	0	3	7	8

- If you need to get a single bin for each value, first use the `unique` and `max` functions to create a vector of bins for each element.

`unique(X)`

0	1	3	7	8	9

`bins = [unique(X), max(X)+1]`

0	1	3	7	8	9	10

Need one extra for the upper bound on the last bin.

- Then give these bins to the histcounts function

`histcounts(X, bins);`

2	1	1	2	3	1