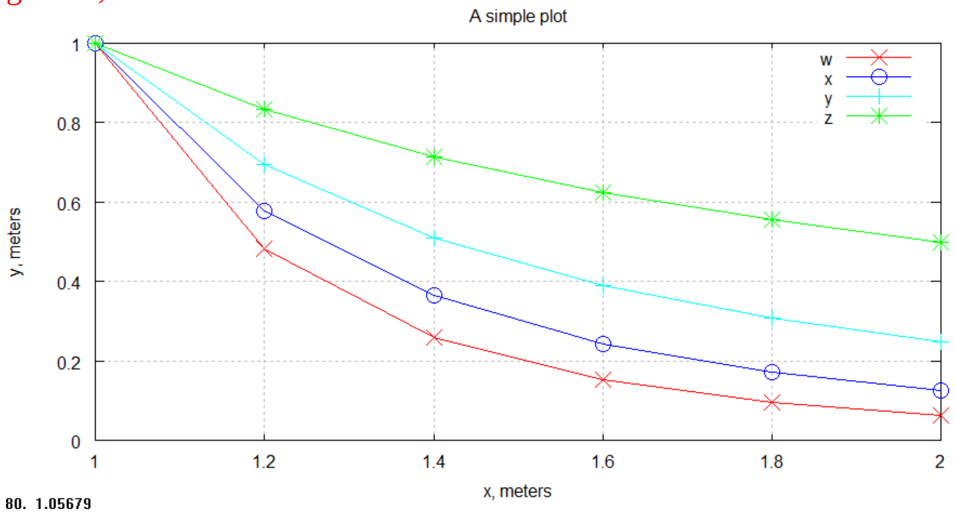
Revisiting multiple plots:

Plotting multiple data sets on one figure using a single **plot( )** function.

In m-file:

```
t = (1 : 0.2 : 2);
w = t.^-4;
x = t.^-3;
y = t.^-2;
z = t.^-1;
plot(t,w, '-xr' , t,x, '-ob' , t,y, '-+c' , t,z, '-*g')
xlabel('x, meters')
ylabel('y, meters')
title('A simple plot')
legend('w' , 'x' , 'y' , 'z')
grid on;
```



You can do the same thing with the **hold on** and **hold off** commands. You only need one **hold on** command.

In m-file:

```
t = (1 : 0.2 : 2);
w = t.^-4;
x = t.^-3;
y = t.^-2;
z = t.^-1;
plot(t,w,'-xr')
hold on
plot(t,x, '-ob')
plot(t,y, '-+c')
plot(t,z, '-*g')
xlabel('x, meters')
ylabel('y, meters')
title('A simple plot')
legend('w' , 'x' , 'y' , 'z')
grid on;
hold off
```

Subplots:

Another way to present multiple sets of data on the same figure is using the **subplot( )** function. This function allows you to subdivide the graphing window into a grid of M rows and N columns.

```
subplot(M,N,p)
```

The variable **p** identifies the position where the subplot will be drawn. The value of **p** is increased in row order. For example, if M and N are both 2, then there will be 4 locations for subplots and then the value of **p** at each of those locations is,

p=1	p=2
p=3	p=4

In m-file:

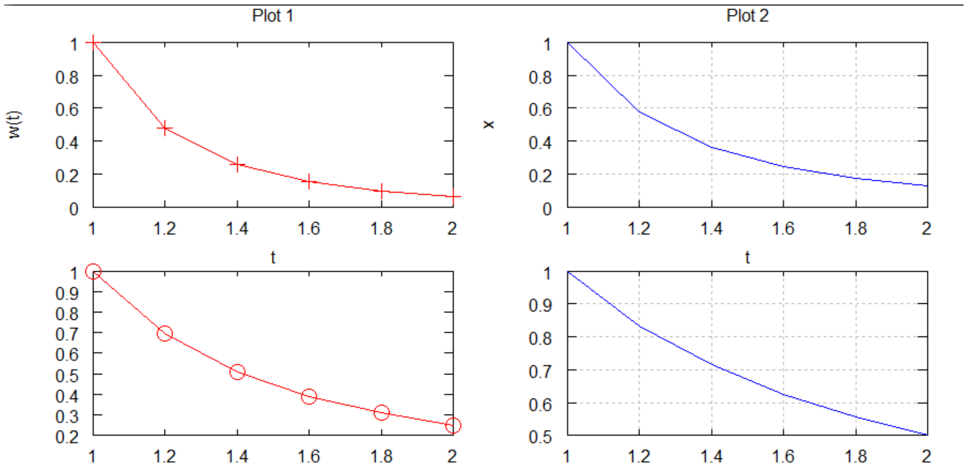
```
t = (1 : 0.2 : 2);
w = t.^-4;
x = t.^-3;
y = t.^-2;
z = t.^-1;

subplot(2,2,1)
plot(t,w)
xlabel('t');
ylabel('w(t)');
title('Plot 1');
grid off;

subplot(2,2,2)
plot(t,x)
xlabel('t');
ylabel('x');
title('Plot 2');
grid on;

subplot(2,2,3)
plot(t,y)
grid off;

subplot(2,2,4)
plot(t,z)
grid on;
```


Plot 1
Plot 2
w(t)
x
t
t
1
0.8
0.6
0.4
0.2
0
1
1.2
1.4
1.6
1.8
2
0.9
0.7
0.5
0.3

Logarithmic axes:

Sometimes data is more conveniently shown on a log scale.

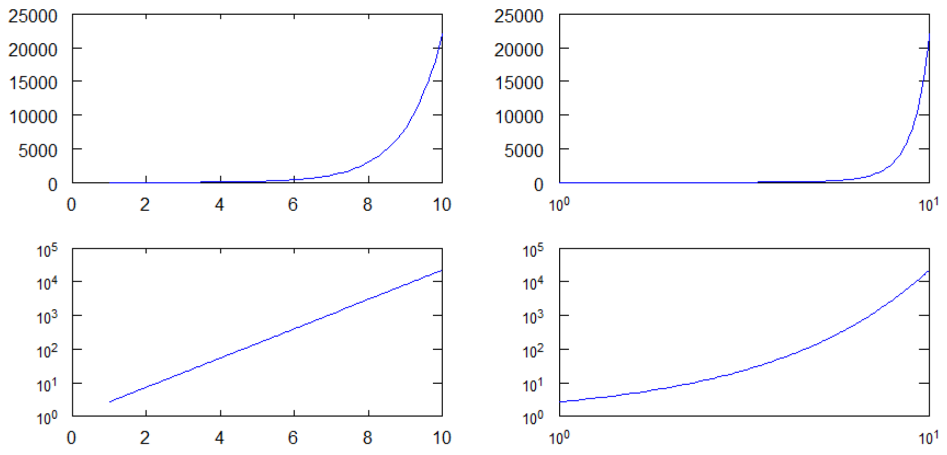
In m-file:

```
x = (0:1:10);
y = exp(x);
subplot(2,2,1)
plot(x,y)

subplot(2,2,2)
semilogx(x,y)

subplot(2,2,3)
semilogy(x,y)

subplot(2,2,4)
loglog(x,y)
```



Histograms:

If you have a lot of similar data that you would like to group into bins (such as test scores), then you can make a histogram to plot the data using the **hist( )** function. Here are a few ways to use this function.

(1) hist(y)

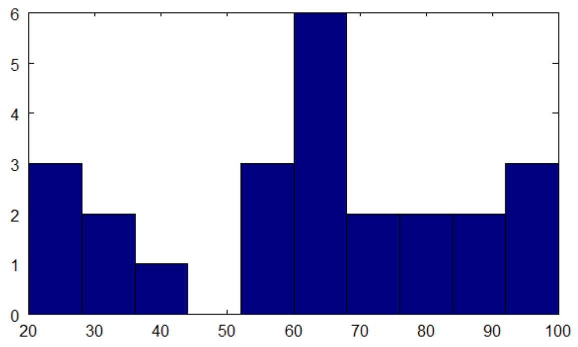
y is an array that contains the data you want to group into bins and plot.

By default, 10 bins will be generated. The leftmost and rightmost edges of the bins will be the minimum and maximum values in the array **y**.

The size of each bin will be (maxval – minval)/10.

In m-file:

```
scores = [20, 99,22,55,66,44,34,65,67, 78,54,76,34,65,54,77, 89,23,65,67,87,76,98,100];
hist(scores)
```



The range of values is 20 to 100.
The bin width is (100-20)/10 = 8.
The bins are centered at 24, 32, 40, 48, 56, 64, 72, 88, 96

If you assign the output from the **hist( )** function to an array, you will store the values in each bin instead of plotting those values on a histogram.

In m-file:

```
scores = [20, 99,22,55,66,44,34,65,67, 78,54,76,34,65,54,77, 89,23,65,67,87,76,98,100];
A = hist(scores);
disp(A)
```

→

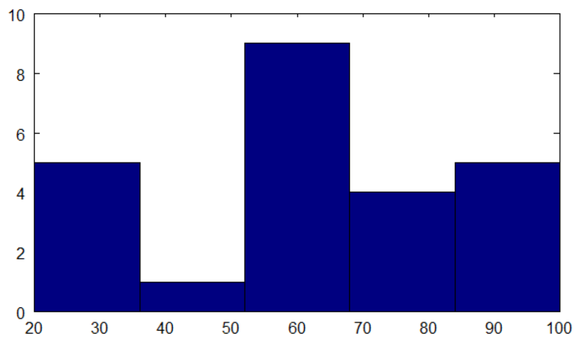
```
  3   2   1   0   3   6   2   2   2   3
```

(2) hist(y, N)

N is the number of bins you will create.

In m-file:

```
scores = [20, 99,22,55,66,44,34,65,67, 78,54,76,34,65,54,77, 89,23,65,67,87,76,98,100];
hist(scores,5);
```



The range of values is 20 to 100.
The bin width is (100-20)/5 = 16.
The bins are centered at 28, 44, 60, 76 , 92

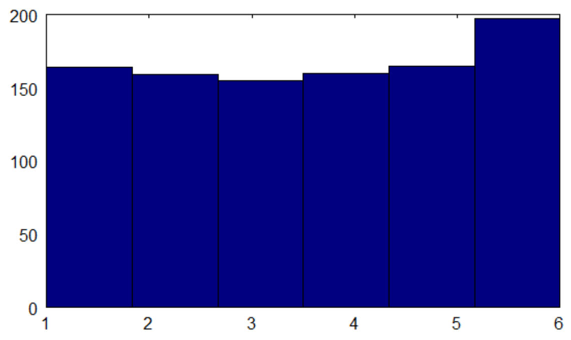
(3) hist(y,x)

You can specify the location of the center of the bins in array **x** as the second argument.

Example: Roll a 6-sided die 1000 times. What is the distribution of values?

In m-file:

```
a = 1;  b = 6;  N = 1000;
randomArray = a + (b-a+1)*rand(1,N);
dice = floor(randomArray);
hist(dice1, 6 )
```



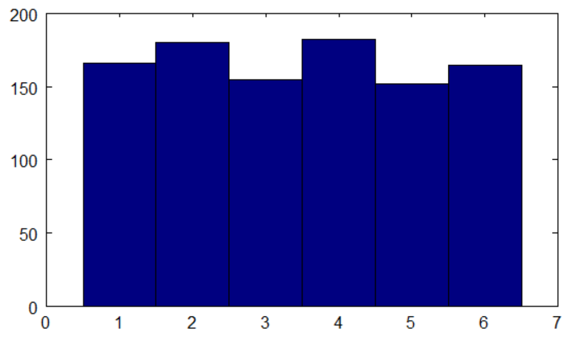
We want 6 bins with a width of 1 for each bin.
However, since the minimum and maximum values are 1 and 6, the width of each bin is (6-1)/6 = 5/6, not 1.

In this case, the center of each bin is ~1.42, 2.25, ~3.08, ~3.92, 4.75, ~5.58

Let’s fix the center of the bins at 1, 2, 3, 4, 5, and 6.

In m-file:

```
a = 1;  b = 6;  N = 1000;
randomArray = a + (b-a+1)*rand(1,N);
dice = floor(randomArray);
x = [1,2,3,4,5,6];
hist(dice1, x )
```



Law of Large Numbers:

Assume that an action (rolling a die) has a theoretical probability distribution (all numbers are equally likely). The more trials (rolls) we conduct, the closer the average of the results will tend toward the expected average.

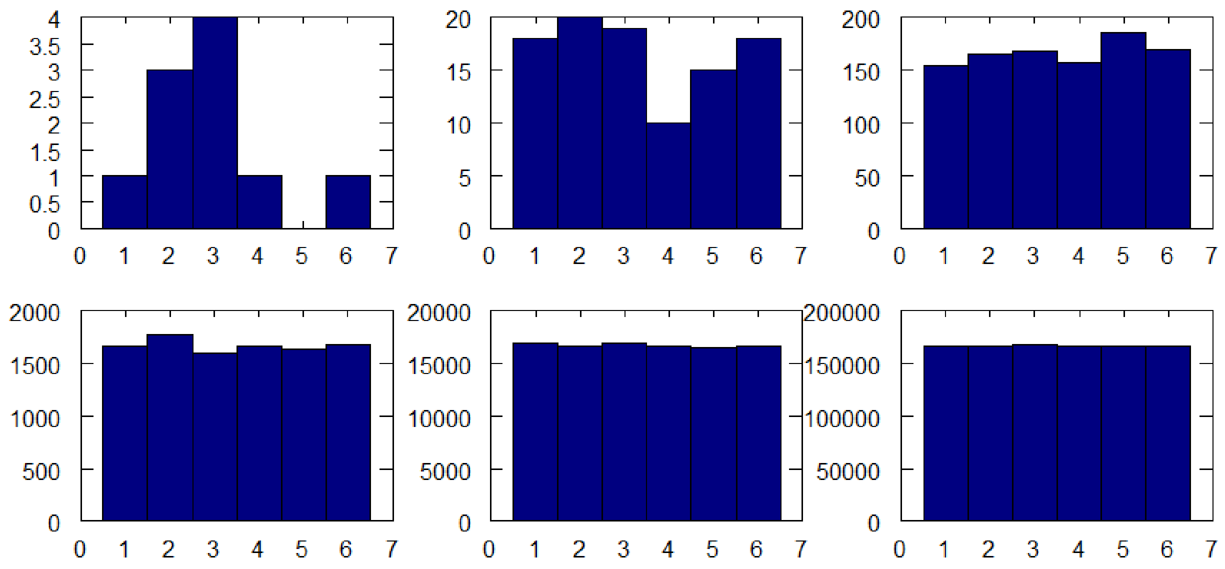
In the case of rolling a 6-sided die, the theoretical probability distribution is uniform for all numbers. The expected average value is,

[(1/6)*1 + (1/6)*2 + (1/6)*3 + (1/6)*4 + (1/6)*5 + (1/6)*6] / 6 = 3.5

Another way of looking at the Law of Large Numbers is that as we increase the number of trials, the closer the results will be to the expected probability distribution when displaying the results on a histogram.

In m-file:

```
a = 1;  b = 6;
die1 = floor( a + (b-a+1)*rand(1,10) );
die2 = floor( a + (b-a+1)*rand(1,100) );
die3 = floor( a + (b-a+1)*rand(1,1000) );
die4 = floor( a + (b-a+1)*rand(1,10000) );
die5 = floor( a + (b-a+1)*rand(1,100000) );
die6 = floor( a + (b-a+1)*rand(1,1000000) );
x = [1,2,3,4,5,6];
subplot(2,3,1)
hist(die1, x )
subplot(2,3,2)
hist(die2, x )
subplot(2,3,3)
hist(die3, x )
subplot(2,3,4)
hist(die4, x )
subplot(2,3,5)
hist(die5, x )
subplot(2,3,6)
hist(die6, x )
```

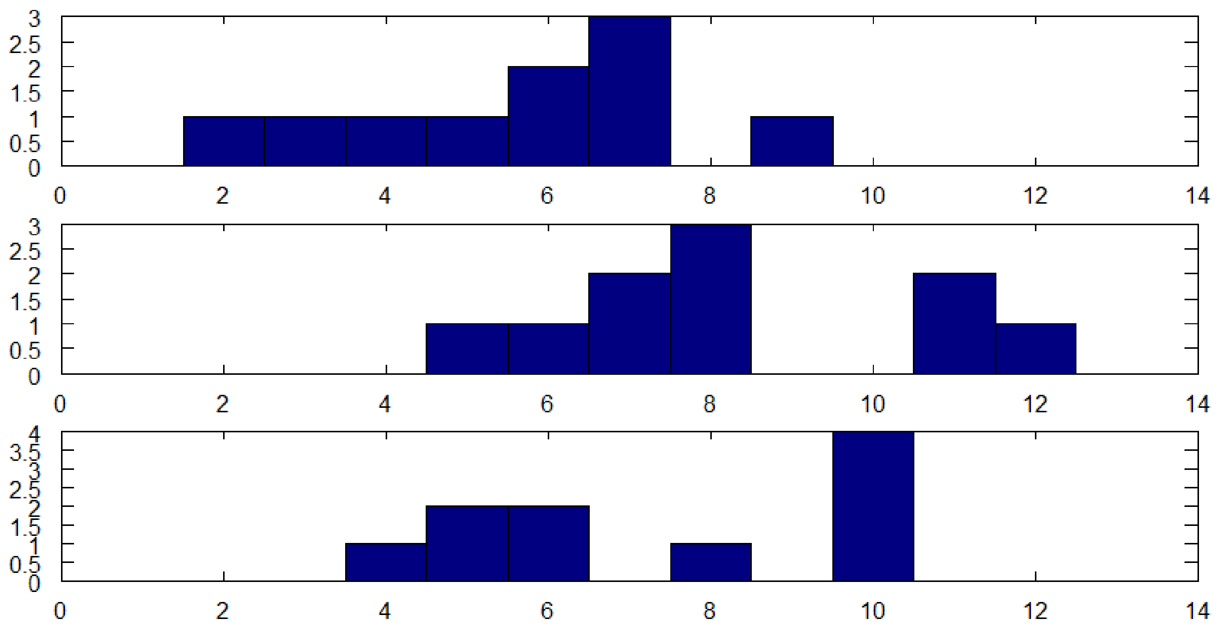

4
3.5
3
2.5
2
1.5
1
0.5
0
20
15
10
5
200
150
100
50
2000
1500
1000
500
20000
15000
10000
5000
200000
150000
100000
50000
0 1 2 3 4 5 6 7

Rolling two dice:

In the casino game Craps, two dice are rolled. The odds of you rolling a 7 is 5-to-1; On average, if you roll the pair of pair of dice 6 times, the values on the pair of dice will total 7 once.

In main.m:

```
a = 1;  b = 6;
die1 = floor(a + (b-a+1)*rand(1,10));
die2 = floor(a + (b-a+1)*rand(1,10));
roll1 = die1 + die2;                    % 10 rolls
die3 = floor(a + (b-a+1)*rand(1,10));
die4 = floor(a + (b-a+1)*rand(1,10));
roll2 = die3 + die4;                    % 10 more rolls
die5 = floor(a + (b-a+1)*rand(1,10));
die6 = floor(a + (b-a+1)*rand(1,10));
roll3 = die5 + die6;                    % 10 more rolls
x = [2:12];
subplot(3,1,1)
hist(roll1, x )
subplot(3,1,2)
hist(roll2, x )
subplot(3,1,3)
hist(roll3, x )
```

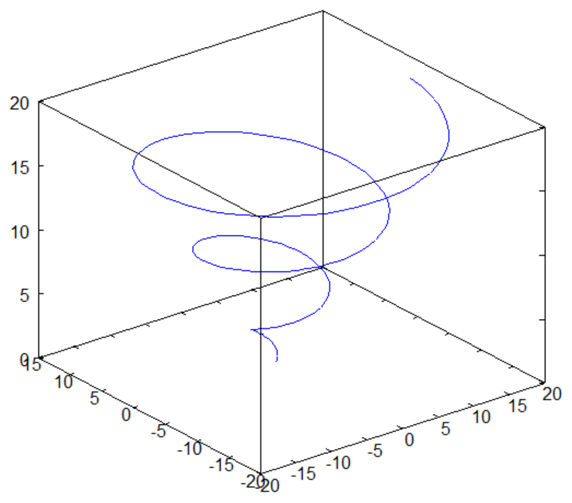


3D Curves:

You can use the **plot3( )** function to generate 3D curves.

In m-file:

```
clear ;clc; clf;
pi = 3.141592654;
t=[0:pi/30:6*pi];
x=t.*cos(t);
y=t.*sin(t);
z=t;
plot3(x,y,z)
```

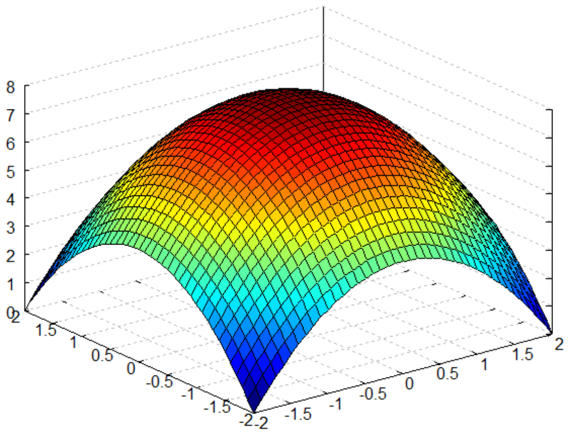


3D Surfaces:

You can use the **surf( )** function to generate 3D surfaces.

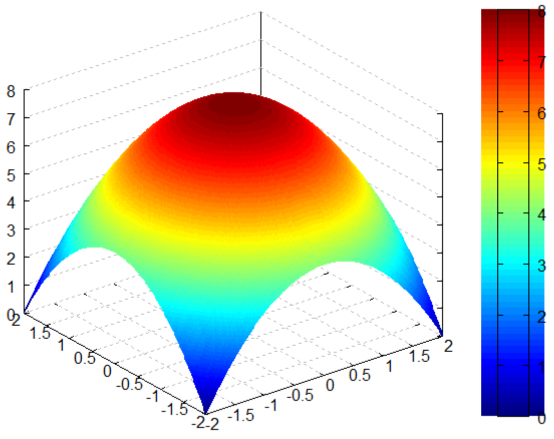
In m-file:

```
clear ;clc; clf;
% A mountain forms a parabolic dome about (0,0)
[x,y]=meshgrid(-2:0.1:2,-2:0.1:2);
surf(x, y,  -x.^2-y.^2 + 8)
```



In m-file:

```
clear ;clc; clf;
 [x,y]=meshgrid(-2:0.1:2,-2:0.1:2);
surf(x,y,  -x.^2-y.^2 + 8,  'EdgeColor', 'none')   % 'EdgeColor','none' removes the gridlines
colorbar              % creates a bar with units
shading interp    % smoothes out the surface
```



8
7
6
5
4
3
2
1
0
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
8
7
6
5
4
3
2
1
0