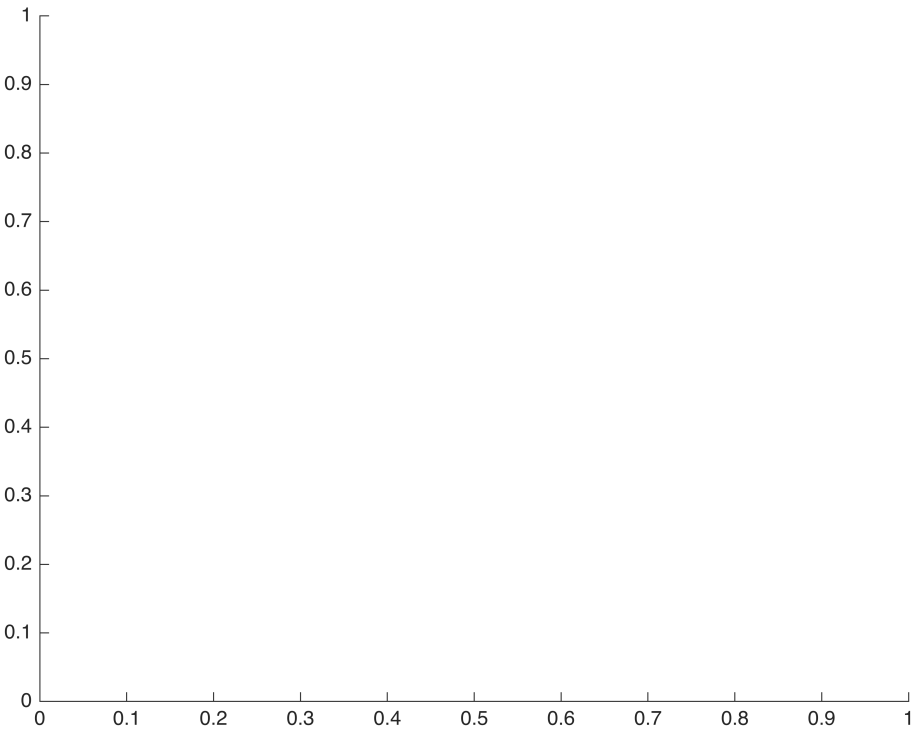
Figures Overview

An introductory guide to MATLAB figures. Demonstrates basic handling and manipulation of figures in MATLAB.


```
% Open a new blank figure window and some axes. Save a handle to the figure
and
% the axes
hfig = figure
```

```
hfig =
  Figure (25) with properties:

      Number: 25
        Name: ''
       Color: [0.94 0.94 0.94]
    Position: [616 498 560 420]
       Units: 'pixels'

  Show all properties
```

```
hax = axes
```




```
hax =
  Axes with properties:

            XLim: [0 1]
            YLim: [0 1]
          XScale: 'linear'
          YScale: 'linear'
    GridLineStyle: '-'
```

```
          Position: [0.13 0.11 0.775 0.815]
             Units: 'normalized'

   Show all properties
```

You will see a summary of the figure and axes properties

You can get and set the values of the properties associated with the handle.

```
get(hfig,'Color')
```

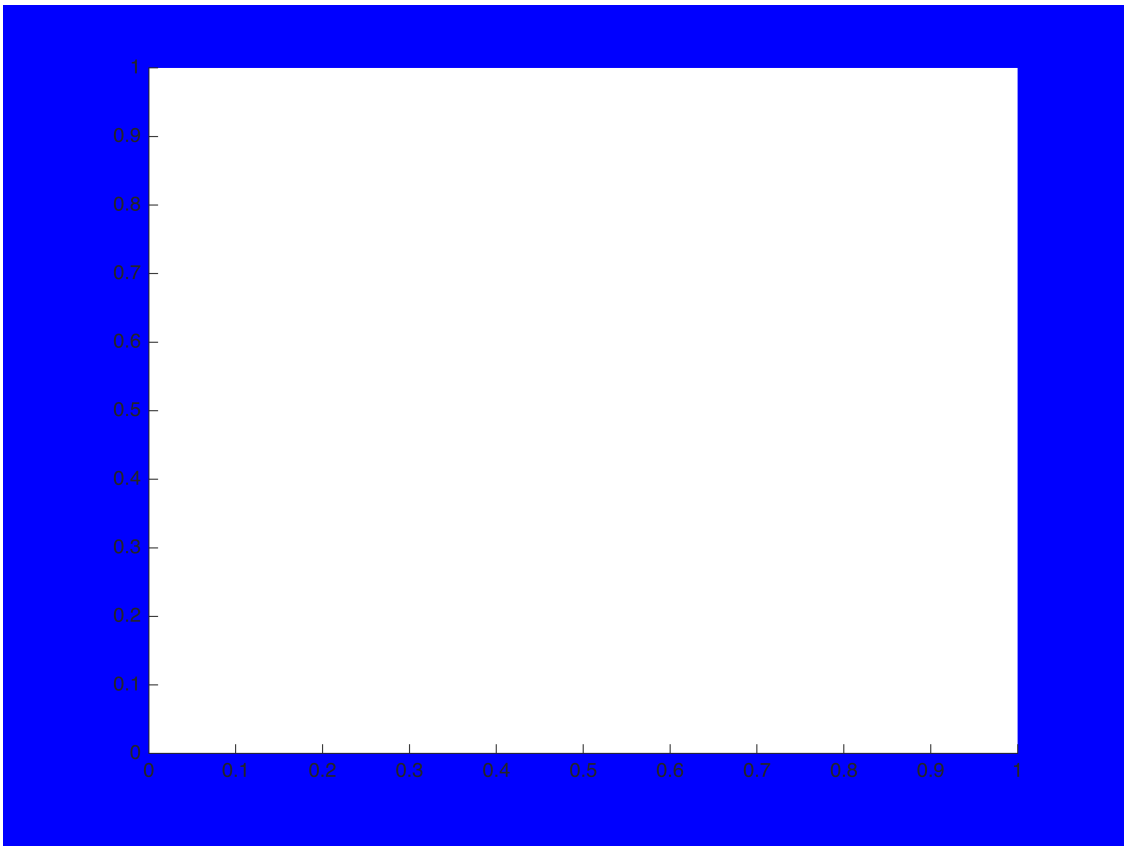
```
ans = 1×3
     1     1     1
```

```
% you can also use dot notation to get the properties
hfig.Color
```

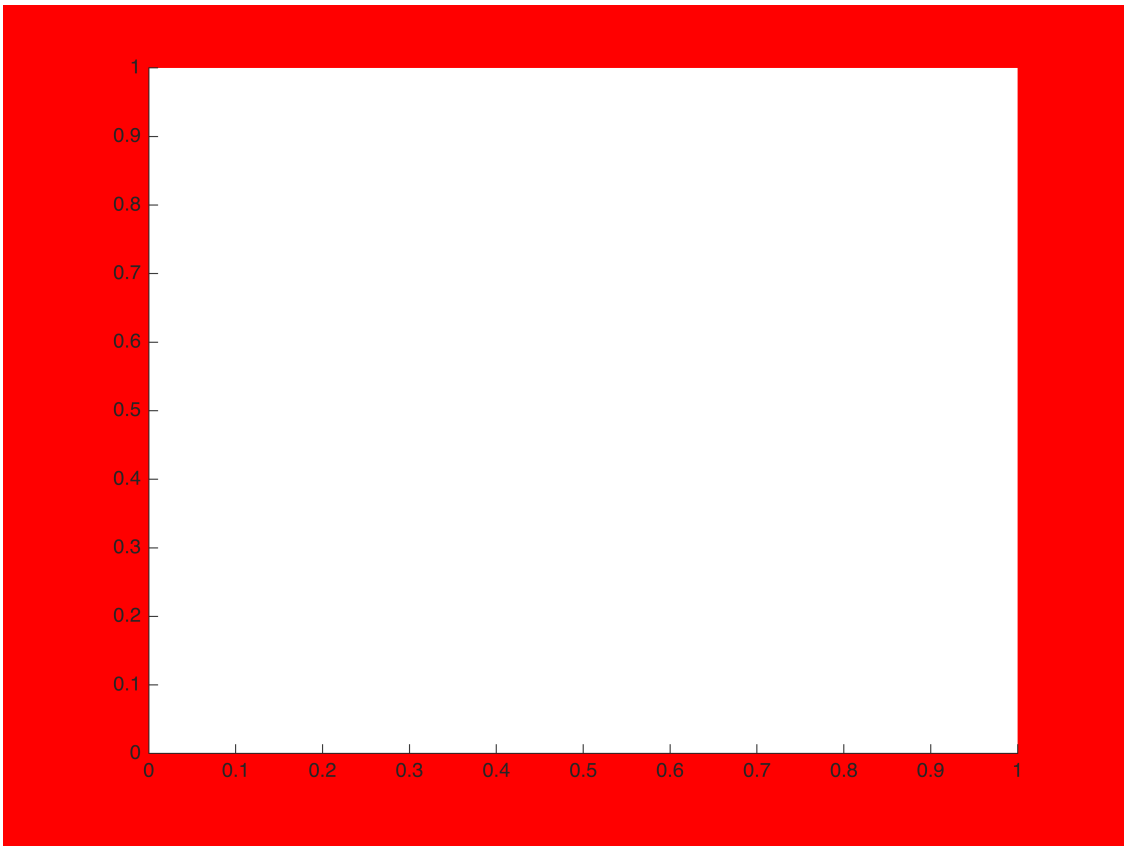
```
ans = 1×3
     1     1     1
```

Note that color here is in RGB units. These vary from 0 (dark) to 1 (bright). For the colour white, there is maximum red, green and blue, i.e. 1 1 1.

```
% Handles can be used to set a property using 'set'
set(hfig,'Color','blue')
```



```
% alternatively, use the dot notation to set a property
hfig.Color = 'red' ;
```



Note that common colours can be specified either using RGB, or just their name. Here 'red' is equvalent to an RGB value of [1 0 0] i.e. all red, no blue, no green.

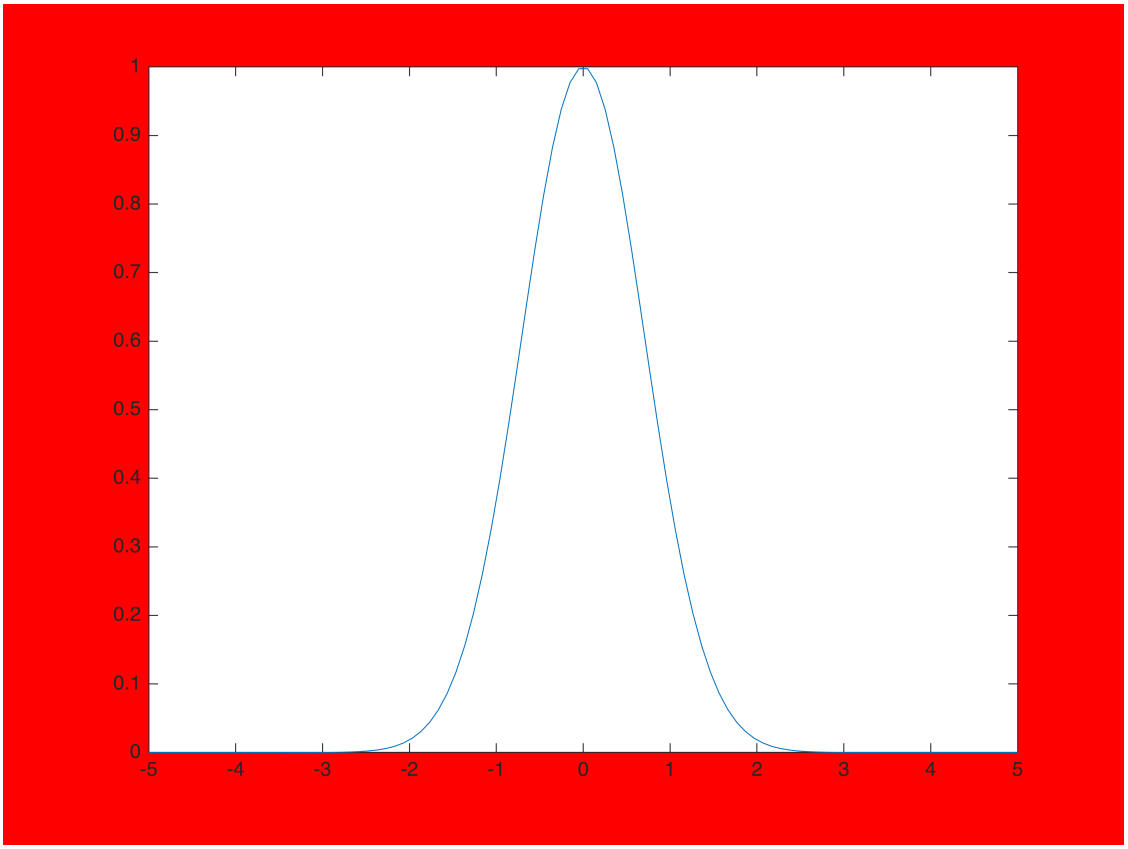
To demonstrate a plot, first create some example data.

Here a normal distribution with zero mean

```
% First set up a vector of 100 numbers, linearly spaced in the range -5 to
+5.
x = linspace(-5, 5, 100) ;

% Calculate the value
y = exp( -x.^2 ) ; % use the '.^2' for element by element squaring of
values in x
```

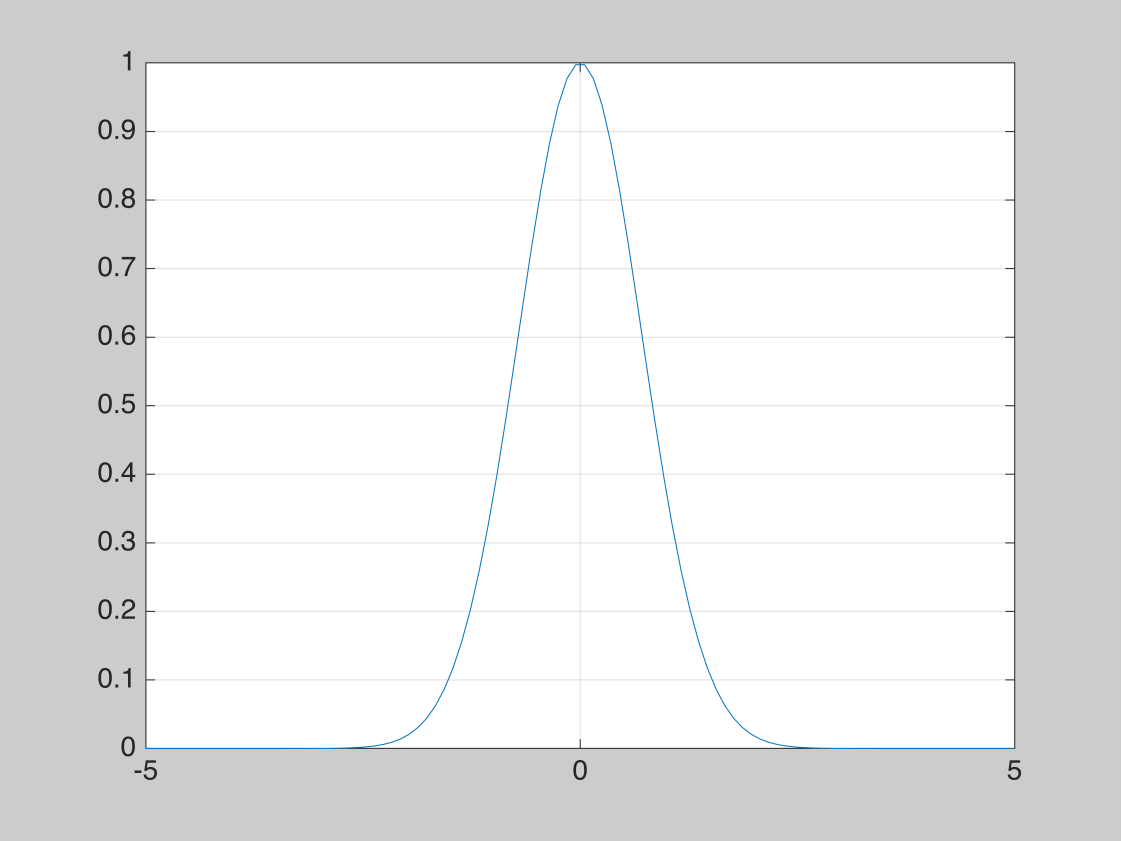
Plot the data

```
plot(x,y)
```



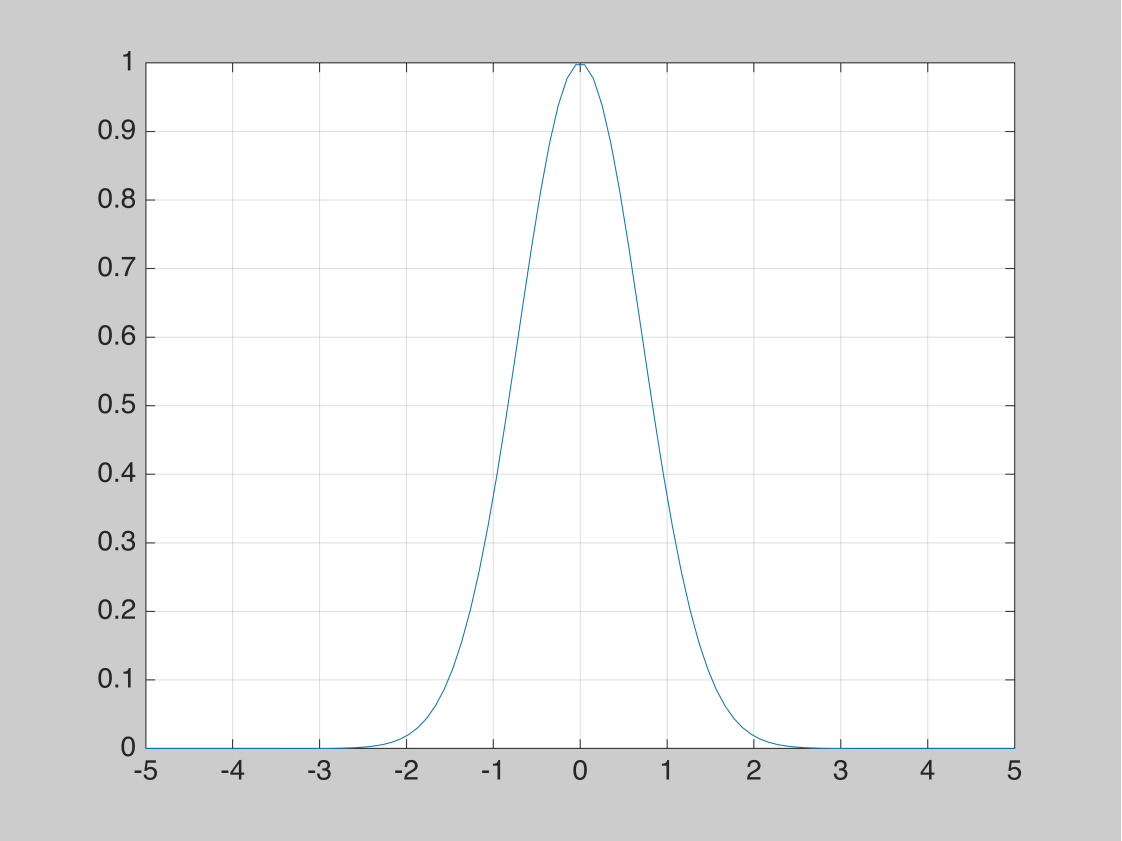
Tidy the plot

```
hax.FontSize = 14 ; % Sets the font size
hfig.Color = [0.8 0.8 0.8] ; % figure back to grey
grid on % place a grid of lines on the axes
```



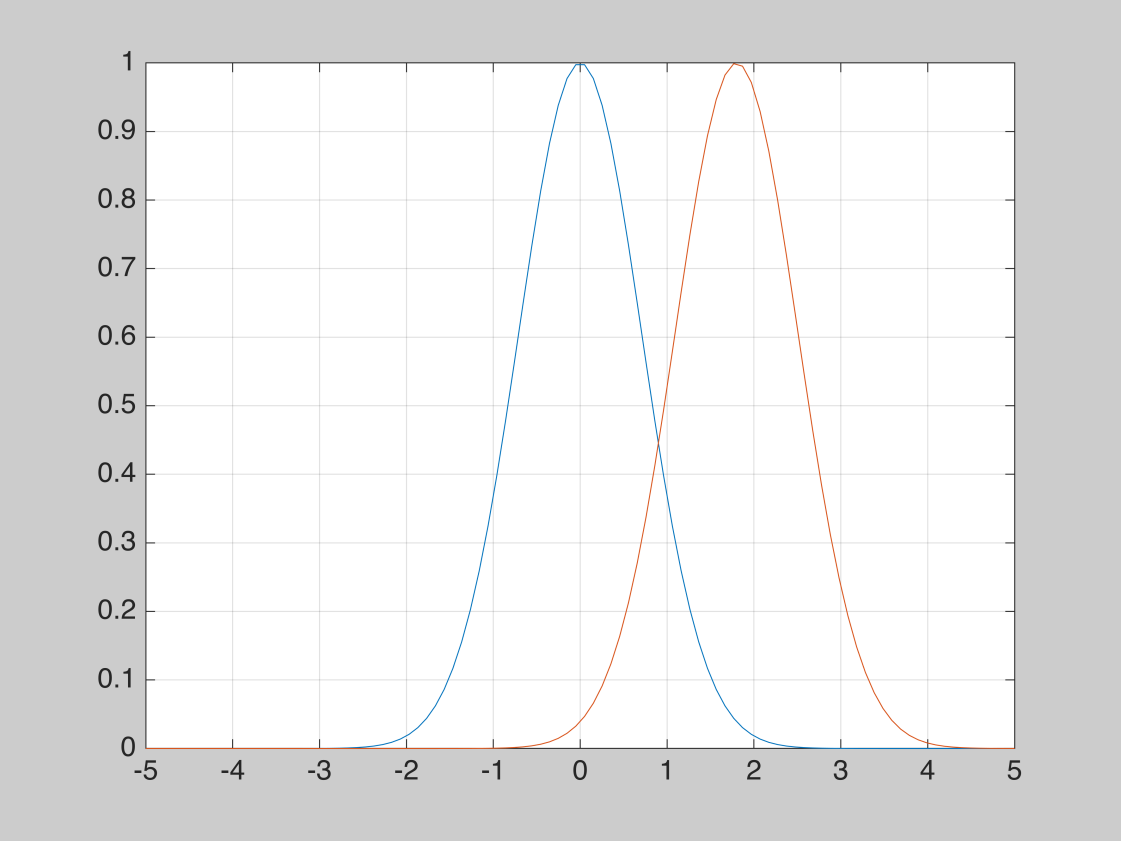
Adjust the x-axis tick marks

```
hax.XTick = [-5 : 5] ; % This produces a vector of numbers from -5 to +5 in
steps of 1.
```



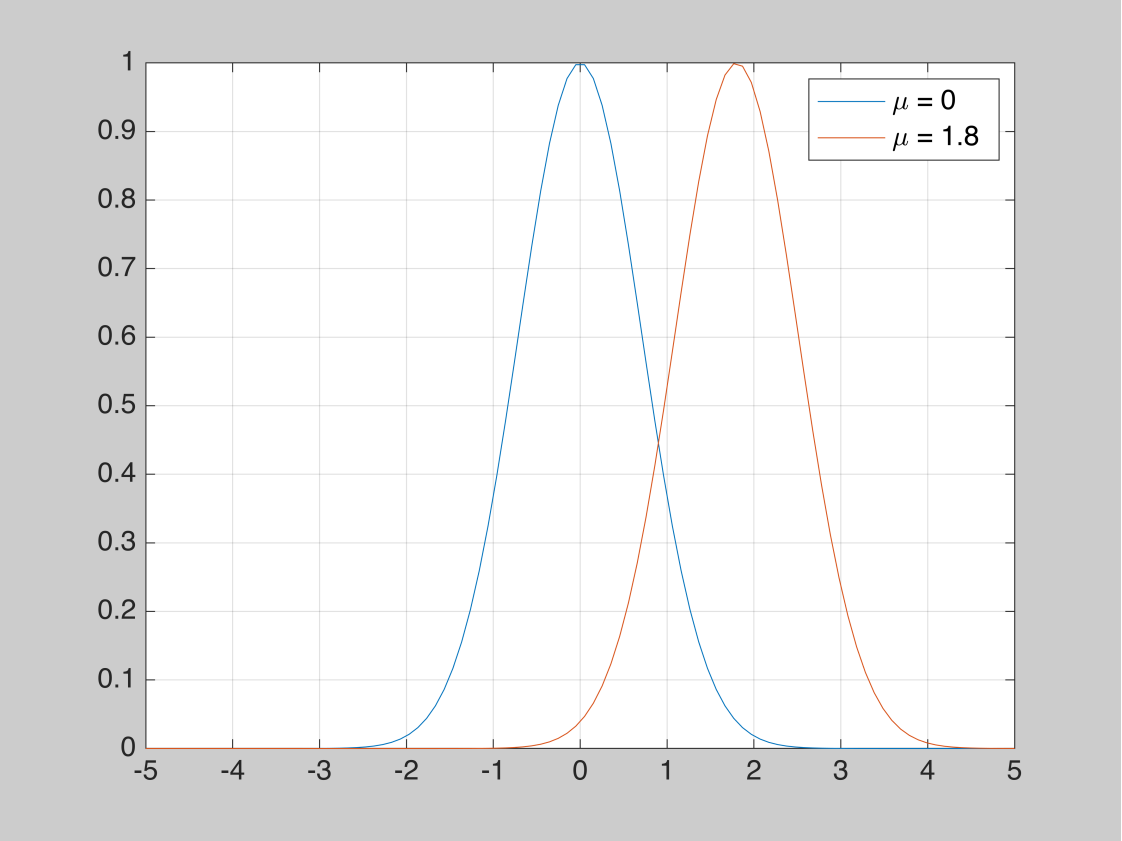
Add a second distribution, this time with a mean of 1.8

```
z = exp( -(x-1.8).^2 ) ;

hold on  % hold keeps the first line in the plot, otherwise it is replaced.

plot(hax, x, z)
```



LaTex formatting is the default

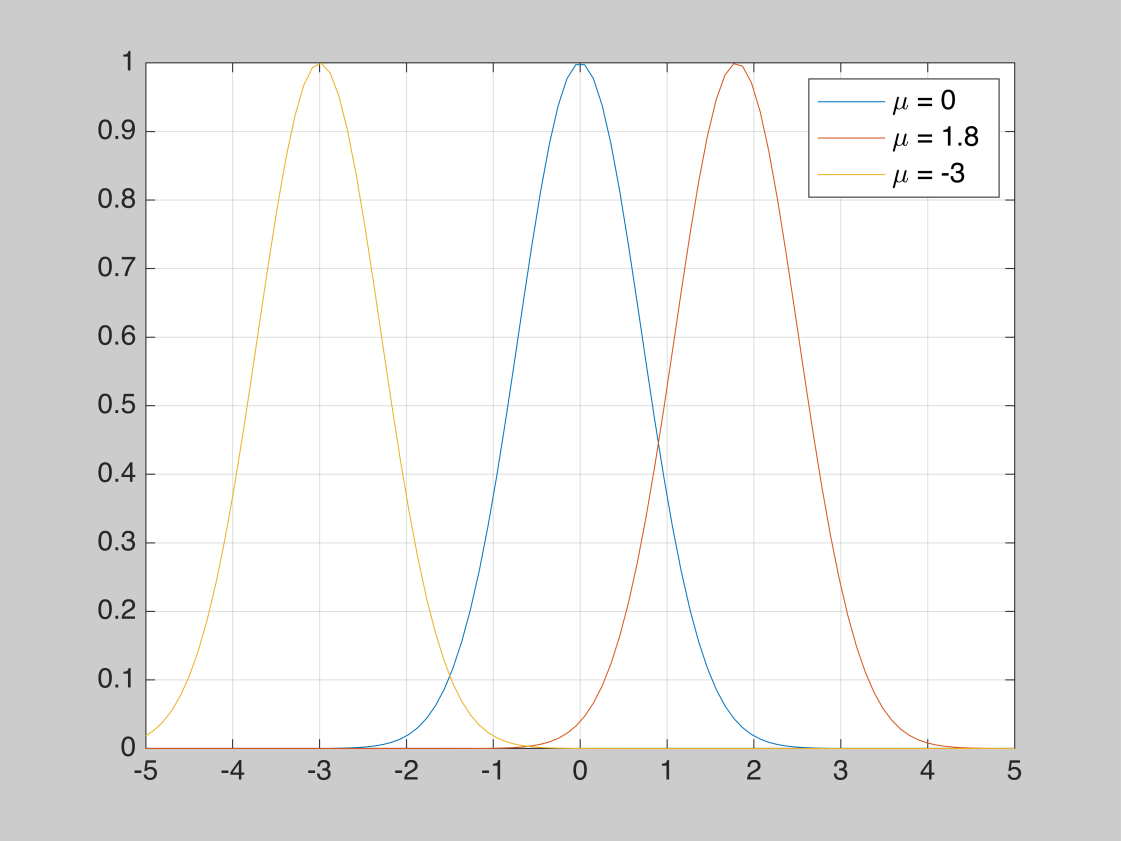
```
legend('\mu = 0', '\mu = 1.8', 'FontSize', 14)
```



To reduce the chance of errors in legends, specify the legend text at time of plotting the line using `DisplayName`.

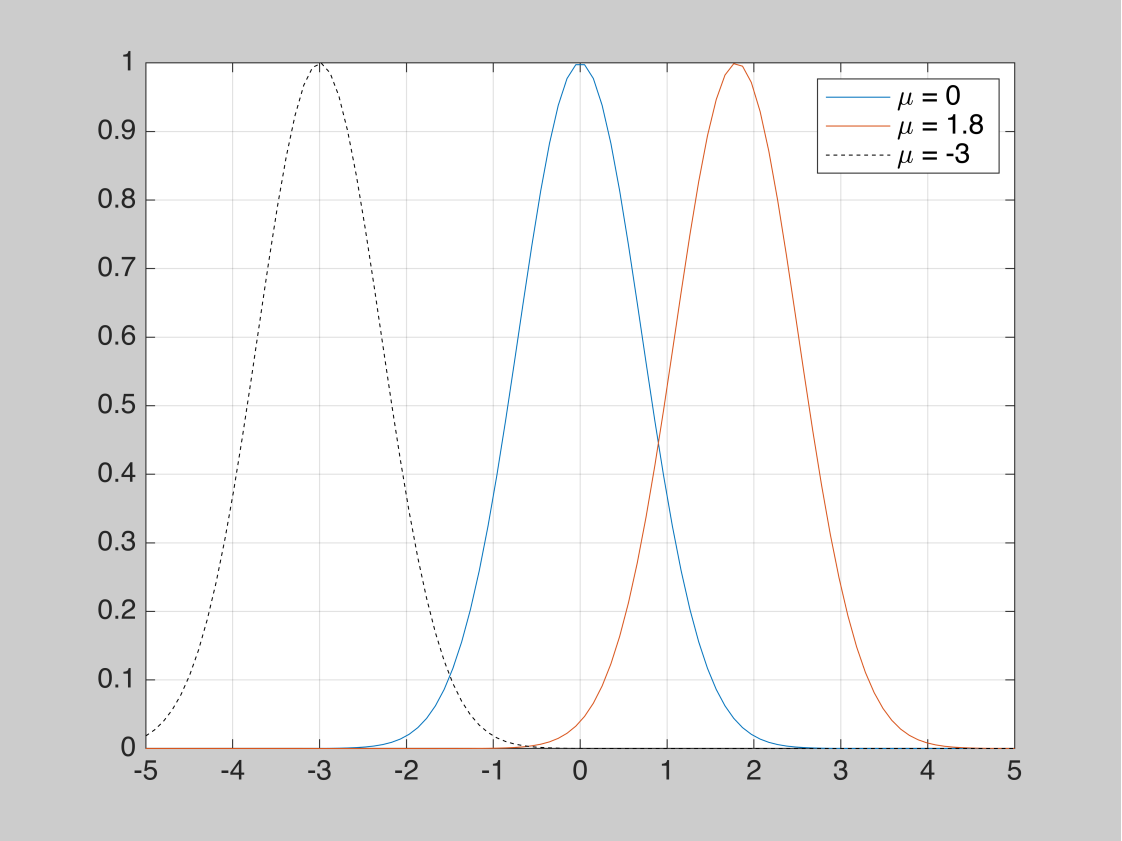
We are also saving a handle here (`hp`) to the line drawn by `plot`

```
w = exp( -(x+3).^2 ) ;

hp = plot(hax, x, w, 'DisplayName','\mu = -3') ;
```



Change the line color (not colour) and style

```
hp.Color = 'black' ;
hp.LineStyle = '--' ;
```



Hopefully this is enough to get started. Getting a figure to look nice needs some trial and error. A typical approach is to inspect the properties associates with the figure, axes or lines, and modify those.

David Atkinson, November 2023.