Plotting How−to

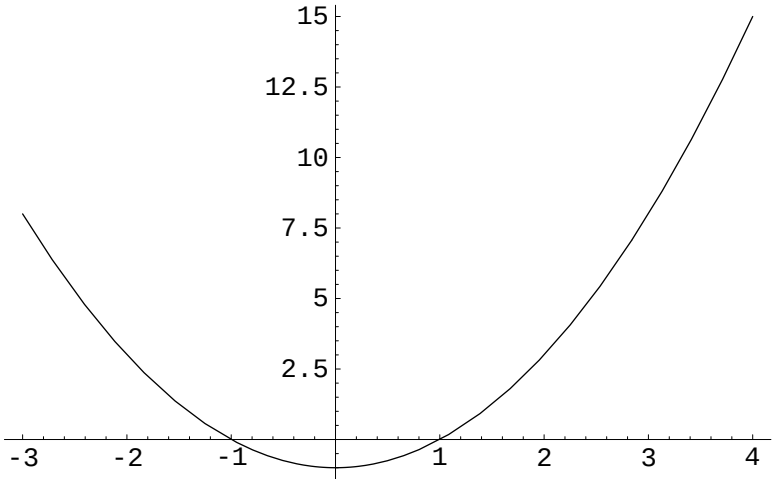
by W. Garrett Mitchener

This worksheet shows a lot of different things you can do with plots, especially how to combine them in different ways. It also shows a couple of techniques for working with lists of points in different forms.

Basic plots

Here's the basic function for plotting functions: `Plot`

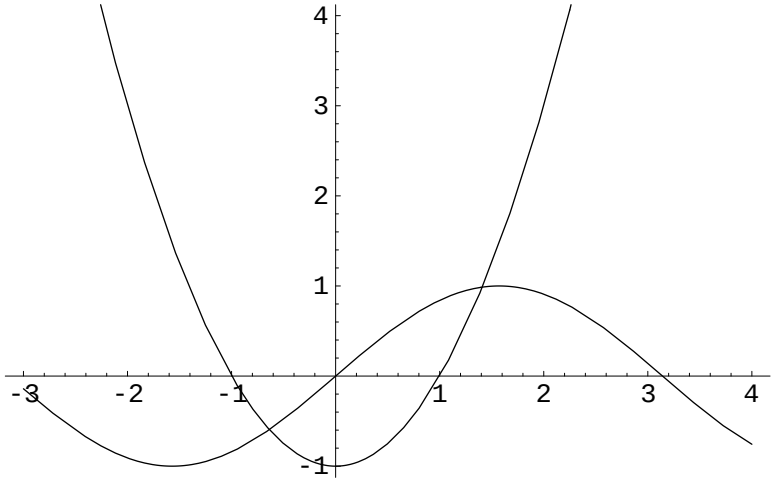
```
Plot[x^2 - 1, {x, -3, 4}]
```




```
- Graphics -
```

You can plot several functions at once:

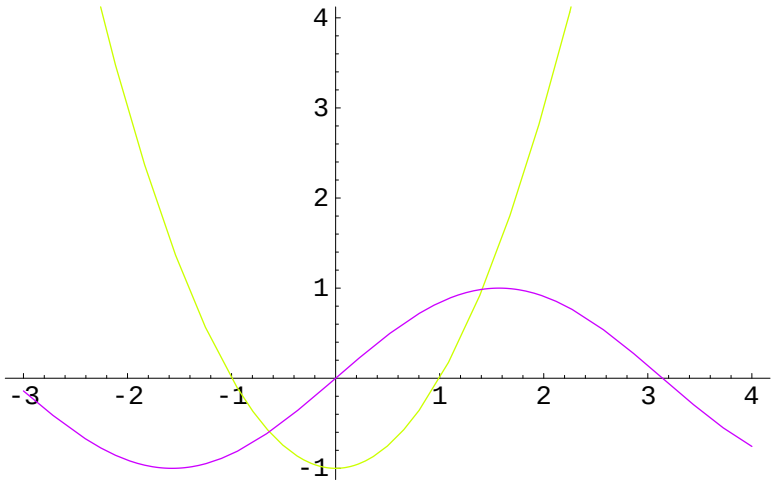
```
Plot[{t^2 - 1, Sin[t]}, {t, -3, 4}]
```




```
- Graphics -
```

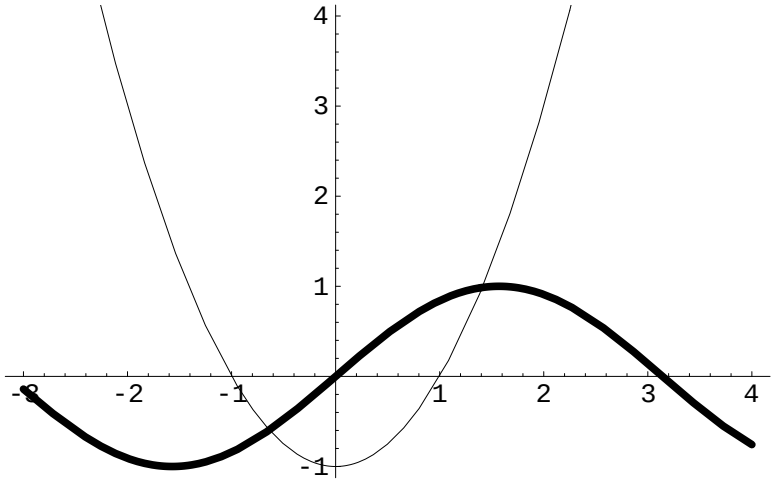
To make the curves different styles, use the option `PlotStyle`.

```
Plot[{x^2 - 1, Sin[x]}, {x, -3, 4},
 PlotStyle → {{Hue[0.2]}, {Hue[0.8]}}]
```




```
- Graphics -
```

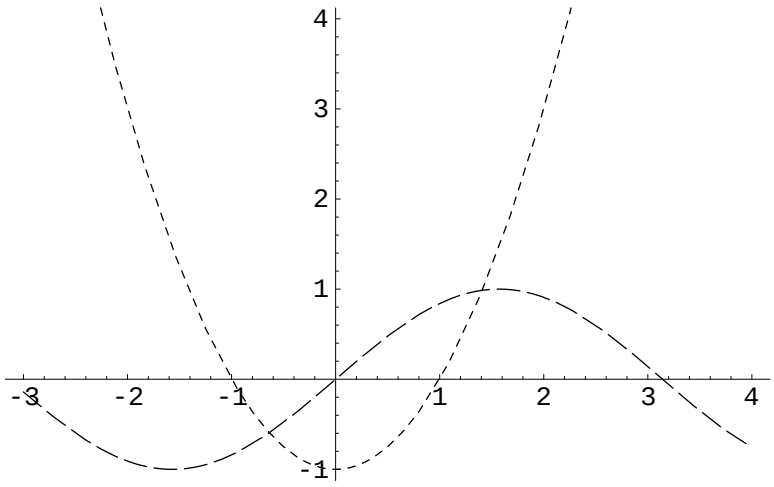
```
Plot[{x^2 - 1, Sin[x]}, {x, -3, 4},
 PlotStyle → {{Thickness[0.001]}, {Thickness[0.01]}}]
```




```
- Graphics -
```

It's good to use `Dashing` or `Thickness` rather than `Hue` because you want the difference to show up clearly when you have to print on a black−and−white printer.

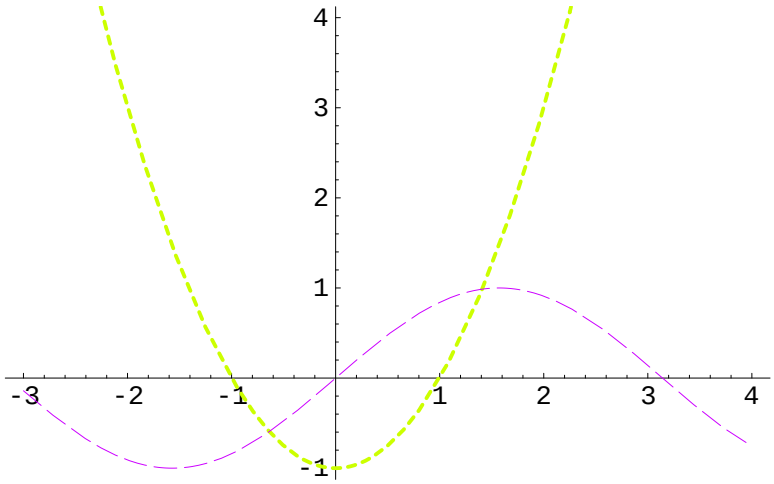
```
Plot[{x^2 - 1, Sin[x]}, {x, -3, 4},
 PlotStyle → {{Dashing[{0.01, 0.01}]}, {Dashing[{0.03, 0.01}]}}]
```




```
- Graphics -
```

You can also combine the styles, which is even better because colors stand out on the screen while you're working or presenting:

```
Plot[{x^2 - 1, Sin[x]}, {x, -3, 4},
 PlotStyle → {{Hue[0.2], Thickness[0.005], Dashing[{0.01, 0.01}]},
   {Hue[0.8], Thickness[0.001], Dashing[{0.03, 0.01}]}}]
```



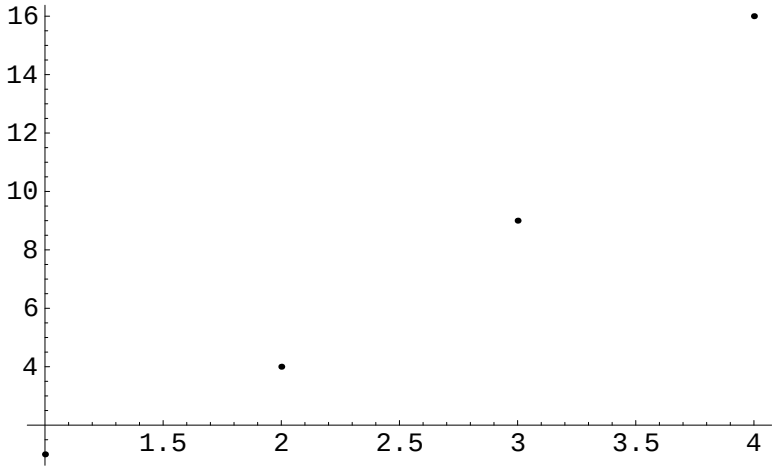

```
- Graphics -
```

If you have point data, you can display it with `ListPlot`.

```
data = {{1, 1}, {2, 4}, {3, 9}, {4, 16}}
```

```
{{1, 1}, {2, 4}, {3, 9}, {4, 16}}
```

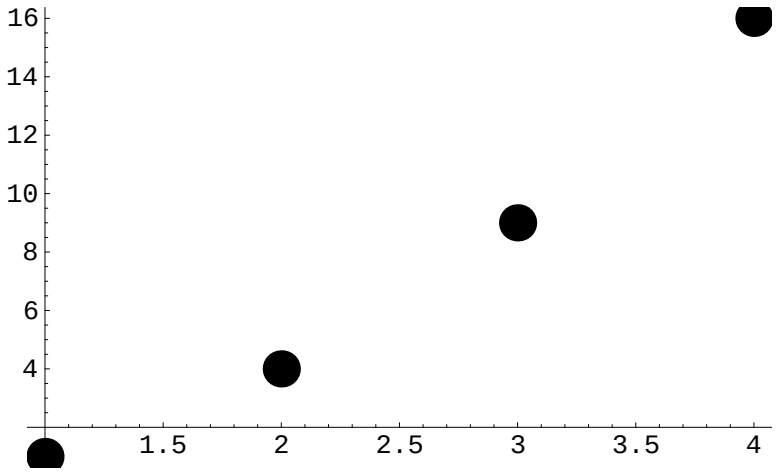
```
ListPlot[data]
```




```
- Graphics -
```

Those tiny dots are hard to see, so try this variation of the `PlotStyle` option:

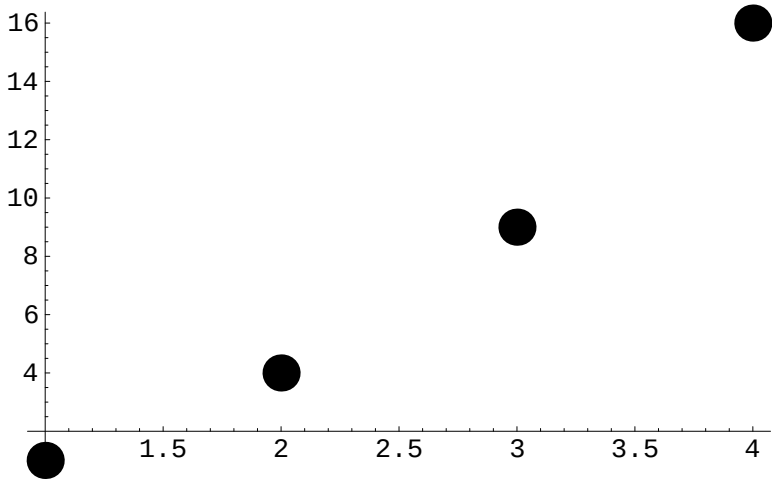
```
ListPlot[data, PlotStyle → PointSize[0.05]]
```




```
- Graphics -
```

Some of our dots got cut off. You can fix that by specifying the `PlotRange` option.

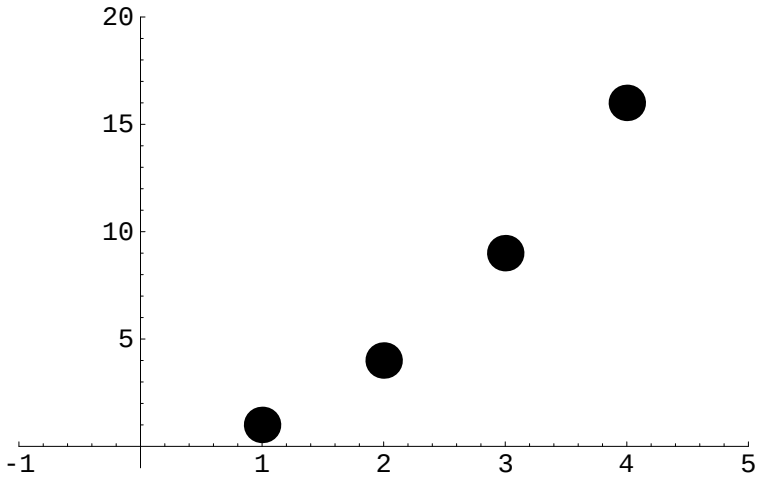
```
ListPlot[data, PlotStyle → PointSize[0.05], PlotRange → All]
```




```
- Graphics -
```

You can also give specific numbers to `PlotRange` in the form {{left, right}, {bottom, top}}. That works for most plot commands, not just ListPlot.

```
ListPlot[data, PlotStyle → PointSize[0.05],
 PlotRange → {{-1, 5}, {-1, 20}}]
```

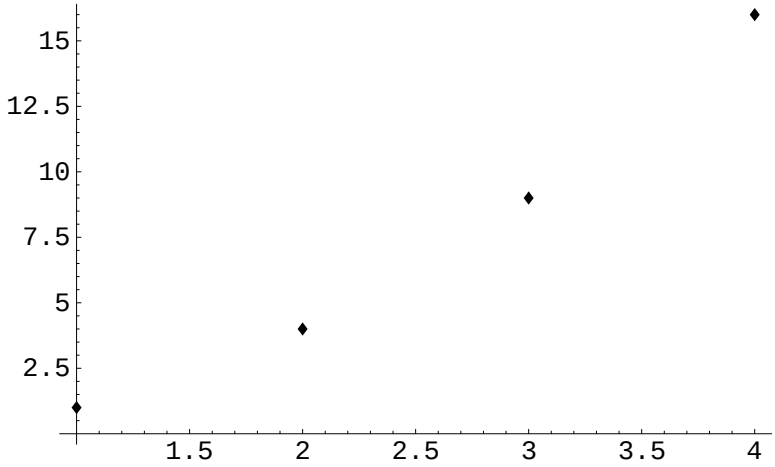



```
- Graphics -
```

There's also a much more powerful function called `MultipleListPlot` that you have to load from a package. The << command loads the Graphics package:

```
<< Graphics'
```

```
MultipleListPlot[{data}]
```



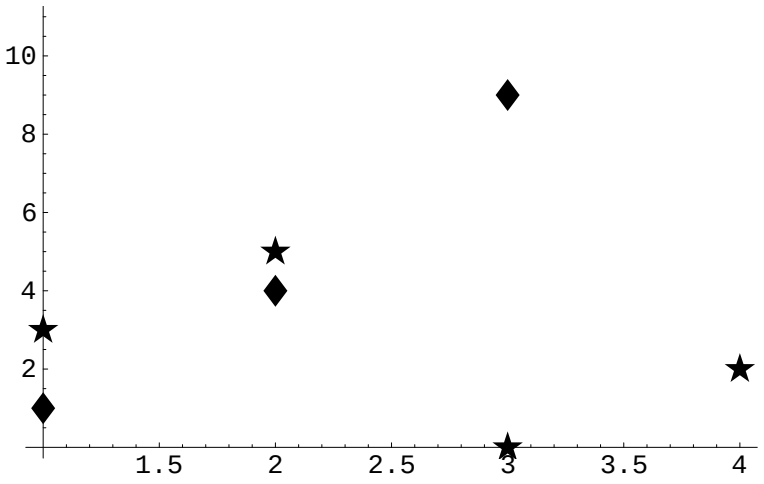

```
- Graphics -
```

```
moreData = {{1, 3}, {2, 5}, {3, 0}, {4, 2}}
```

```
{{1, 3}, {2, 5}, {3, 0}, {4, 2}}
```

Here you can see that MultipleListPlot uses different symbols for each list of points. This is what you have to do to make them larger:

```
MultipleListPlot[{data, moreData},
 SymbolShape → {PlotSymbol[Diamond, 6], PlotSymbol[Star, 6]}]
```




```
- Graphics -
```

How to turn a pair of lists into a list of pairs

```
xValues = {1, 2, 3}
```

```
{1, 2, 3}
```

(This should give you a spelling warning−− Don't worry about it, it's harmless.)

```
yValues = {0, 3, 8}
```

```
– General::spell1 : Possible spelling error: new symbol
    name "yValues" is similar to existing symbol "xValues". More...
```

```
{0, 3, 8}
```

This expression creates a matrix whose two rows are xValues and yValues.

```
{xValues, yValues}
```

```
{{1, 2, 3}, {0, 3, 8}}
```

You can see that with the `MatrixForm` command.

```
{xValues, yValues} // MatrixForm
```

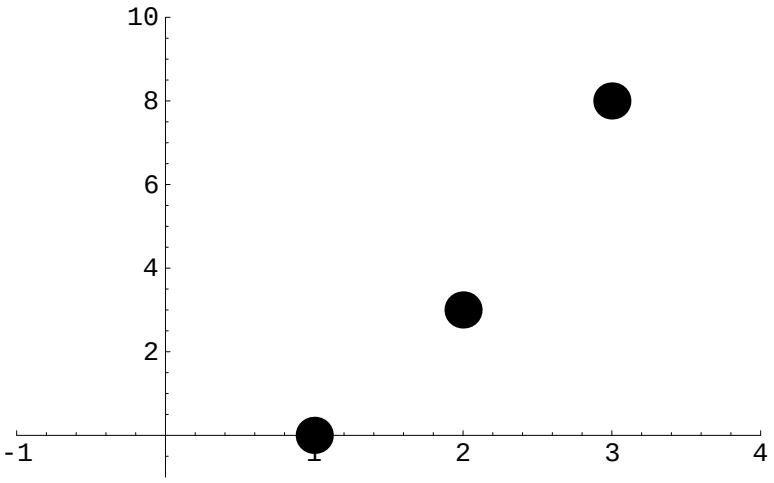
$$\begin{pmatrix} 1 & 2 & 3 \\ 0 & 3 & 8 \end{pmatrix}$$

We want two columns instead of two rows, so all we have to do is take the transpose.

```
points = Transpose[{xValues, yValues}]
```

```
{{1, 0}, {2, 3}, {3, 8}}
```

```
ListPlot[points, PlotStyle → PointSize[0.05],
 PlotRange → {{-1, 4}, {-1, 10}}]
```




```
- Graphics -
```

How to combine plots

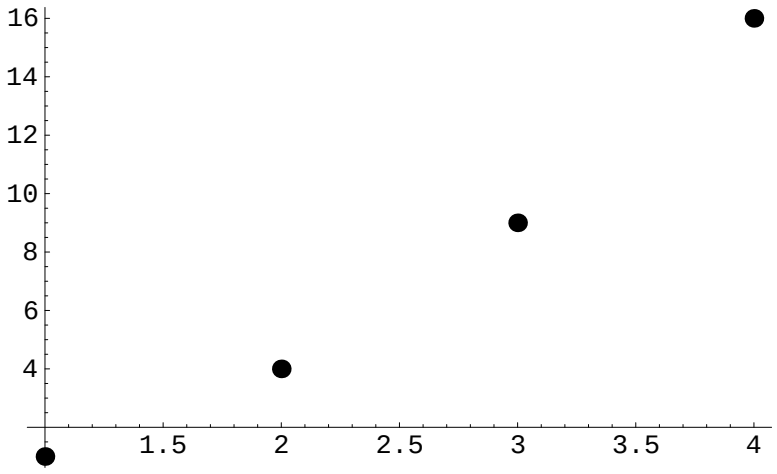
Suppose you want to superimpose two plots, say one with a list of points and one with a function. Neither the `Plot` function nor the `MultipleListPlot` function can do this. `Plot` can display any number of functions, and `MultipleListPlot` can display any number of lists of points, but you can't mix the two directly. Instead, you can save the actual graphics objects in a variable and use a special graphics function to show several of them at the same time. Here's an example.

Let's continue with the data we defined earlier:

```
data
```

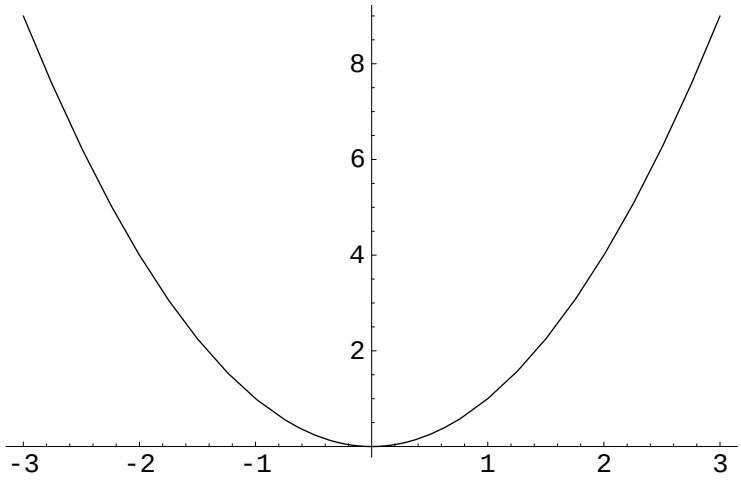
```
{{1, 1}, {2, 4}, {3, 9}, {4, 16}}
```

```
pointPlot = ListPlot[data, PlotStyle → PointSize[0.025]]
```




```
- Graphics -
```

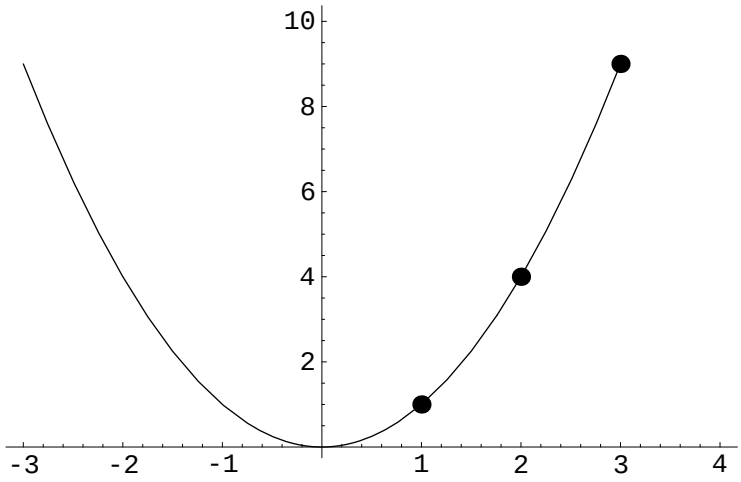
```
functionPlot = Plot[x^2, {x, -3, 3}]
```




```
- Graphics -
```

Now use Show to combine the two:

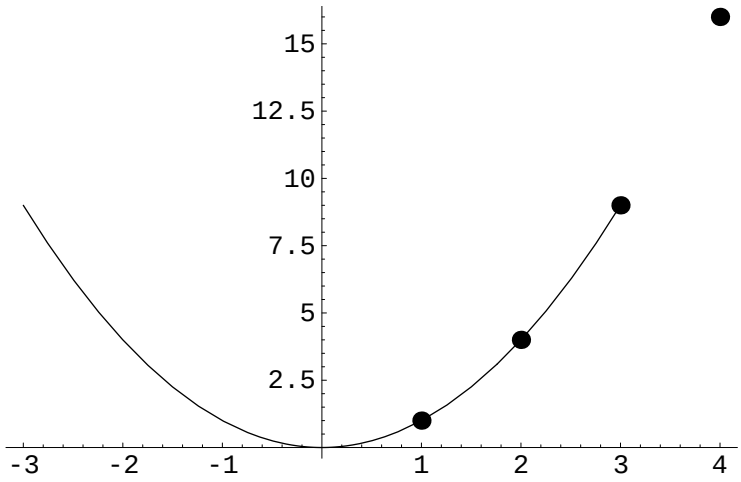
```
Show[functionPlot, pointPlot]
```




```
- Graphics -
```

One of our points got cut off. That's because `Show` has to combine the options given to the different plots, so sometimes it doesn't get the range correct, and so forth. To fix it, we just add a `PlotRange` option to `Show`

```
Show[functionPlot, pointPlot, PlotRange → All]
```



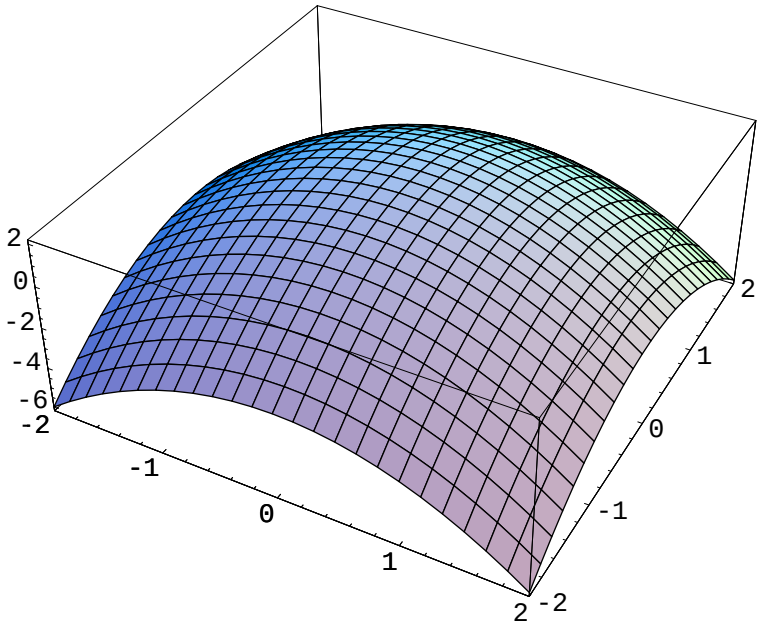

```
- Graphics -
```

The curve doesn't go all the way to that last point because we specified a range of {−3,3} when we created functionPlot.

3−Dimensional plotting

Mathematica can also plot surfaces. Here's an example of the `Plot3D` command:

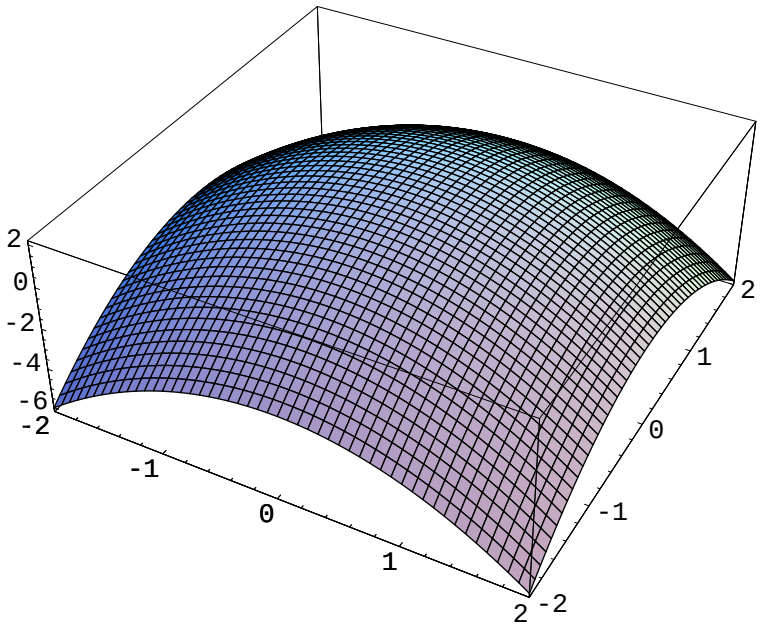
```
Plot3D[2 - x^2 - y^2, {x, -2, 2}, {y, -2, 2}]
```




```
- SurfaceGraphics -
```

The surface is approximated by small flat faces. There are zillions of options you can add to a surface plot. One of the more useful ones is to control how fine the mesh is with the `PlotPoints` option:

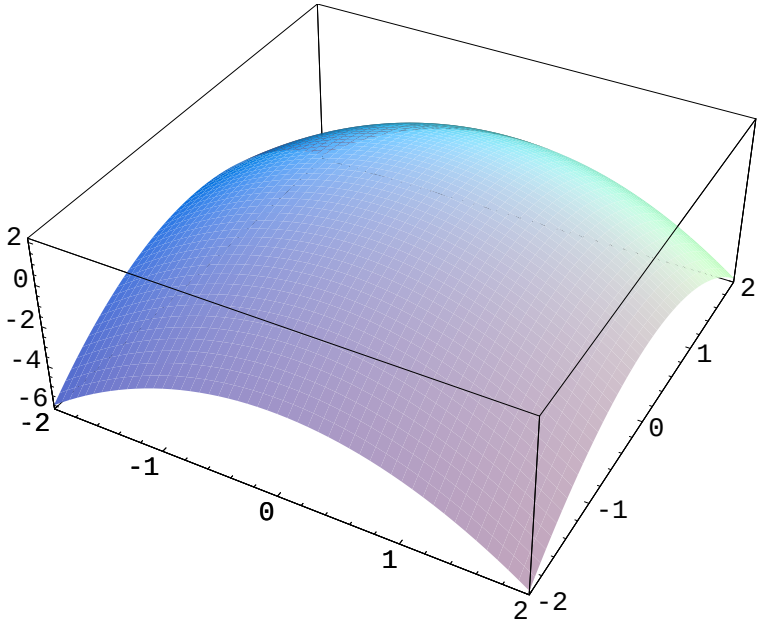
```
Plot3D[2 - x^2 - y^2, {x, -2, 2}, {y, -2, 2}, PlotPoints → 50]
```




```
- SurfaceGraphics -
```

If you use a fine mesh, the mesh lines can get distracting, so you might want to turn them off by specifying the `Mesh` option:

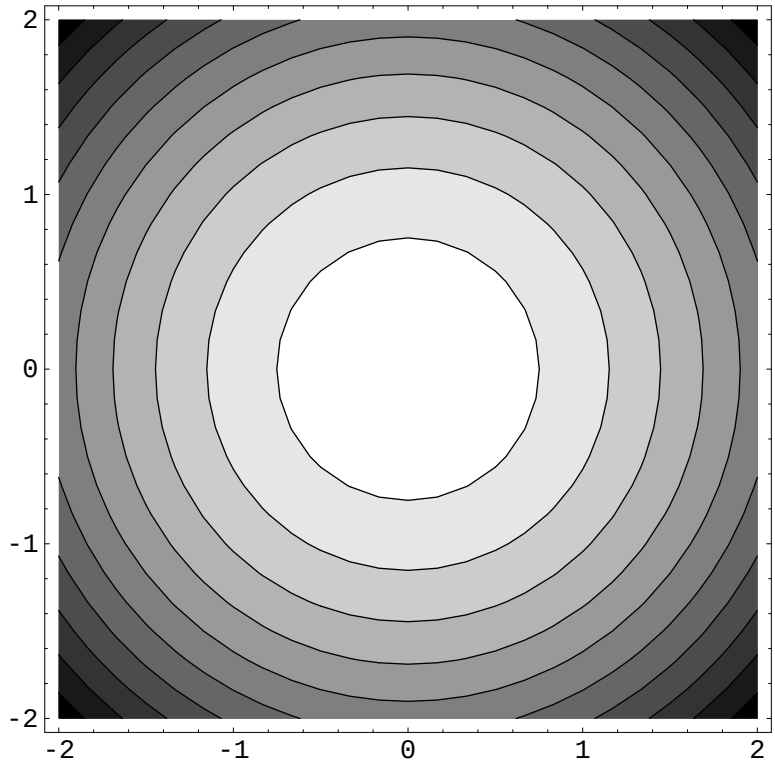
```
Plot3D[2 - x^2 - y^2, {x, -2, 2},
 {y, -2, 2}, PlotPoints → 50, Mesh → False]
```




```
- SurfaceGraphics -
```

Sometimes, a surface isn't the best way to view a function of two variables. You can also use `ContourPlot` which draws and colors level sets of the function.

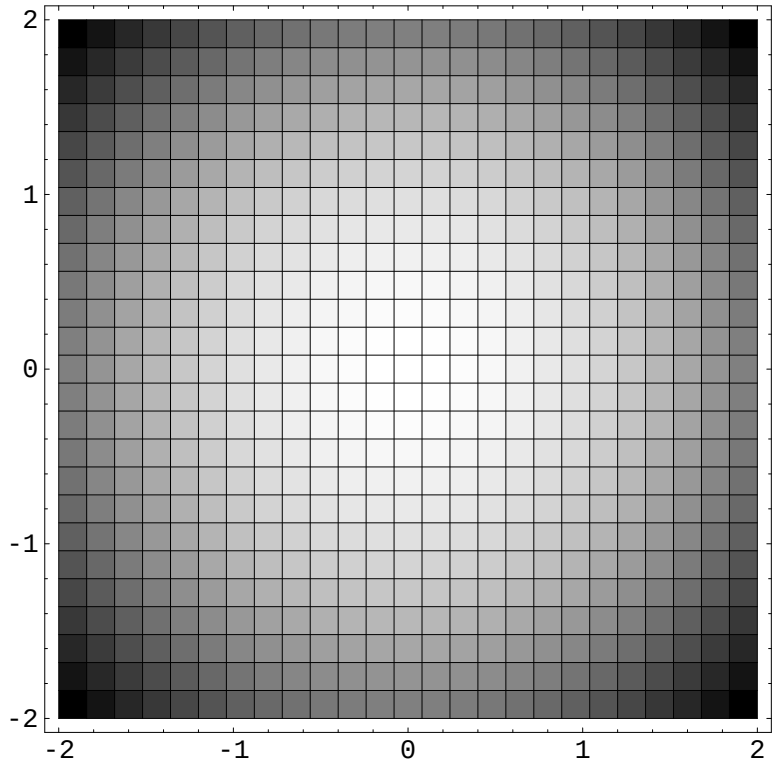
```
ContourPlot[2 - x^2 - y^2, {x, -2, 2}, {y, -2, 2}]
```




```
- ContourGraphics -
```

And there's also `DensityPlot`, which splits the plane into boxes and colors each box based on the function value there.

```
DensityPlot[2 - x^2 - y^2, {x, -2, 2}, {y, -2, 2}]
```



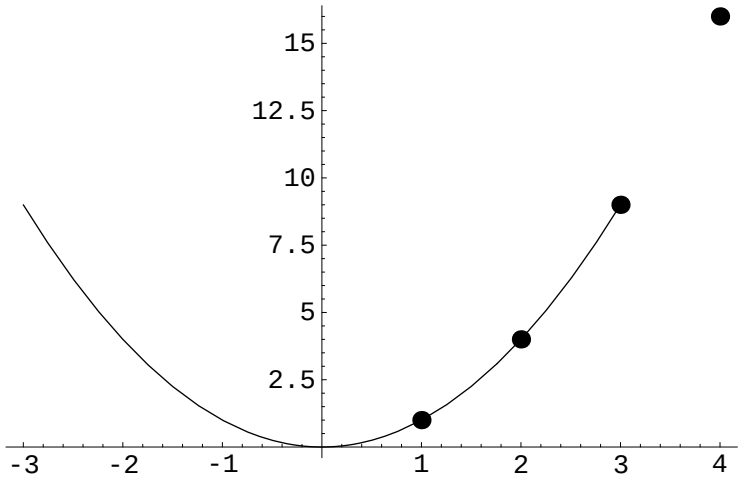

```
- DensityGraphics -
```

Saving plots for use in LaTeX and other programs

What if you want to save a picture to use in another program? One way is to use the `Export` function. It takes a file name in quotes and an object to export. The format of the exported file is determined by the extension on the file name, for example, Picture.png is a PNG file, Picture.jpg is a JPEG file, etc.

Let's go back to the composite plot from before:

```
compositePlot = Show[functionPlot, pointPlot, PlotRange → All]
```




```
- Graphics -
```

If you want to save this for LaTeX, the best format is Encapsulated Post Script (or EPS for short). EPS files are made up of drawing commands to be interpreted by a printer, and they can be drawn at any resolution. Most journals want EPS files for all graphics in papers since they usually print their issues in at least 1200 dpi.

```
Export["CompositePlot.eps", compositePlot]

CompositePlot.eps
```

Then you can load it into a LaTeX file with commands like this:

```
\documentclass{article}
\usepackage{graphicx}

\begin{document}

\includegraphics{CompositePlot.eps}

\end{document}
```

And the \includegraphics command takes options like

```
\includegraphics[width=1in]{...}
\includegraphics[height=1in]{...}
```

If you want to save this for the web or to put in a Word, Word Perfect, or Open Office document, a good format is Portable Network Graphics, or PNG. This is a compressed, pixel−based graphics format, and *Mathematica* will render the graphics at a specific resolution. Other formats that are good for the web are JPEG (which is more highly compressed because fine details are left out) and GIF (which is an older format similar to PNG).

```
Export["CompositePlot.png", compositePlot]

CompositePlot.png
```

If you want to edit the graphics after you've saved them, say to add arrows or other decorations, you have several options. For EPS files, Adobe Illustrator is great for editing them directly. XFig and Sketch and other free UNIX programs can import them, and then you can save the resulting figure as EPS. For PNG and other pixel−based formats, try a paint program like Adobe Photoshop or the GIMP. (Many paint programs can open EPS

files, but they convert them to pixel data first, so you lose the ability to print them at any resolution.) Word, Word Perfect, and Open Office can also do some drawing on top of imported graphics.

`Export` also has lots of options, but you probably won't need them unless you get into really fancy graphics.