Data Plots

There is a new dataplot command for plotting numerical data. It is similar in scope to the plot and plot3d commands but is specifically designed for displaying data. It encompasses both 2-D and 3-D plots. The dataplot command is also available in the right-click context-sensitive "Plots" submenu.

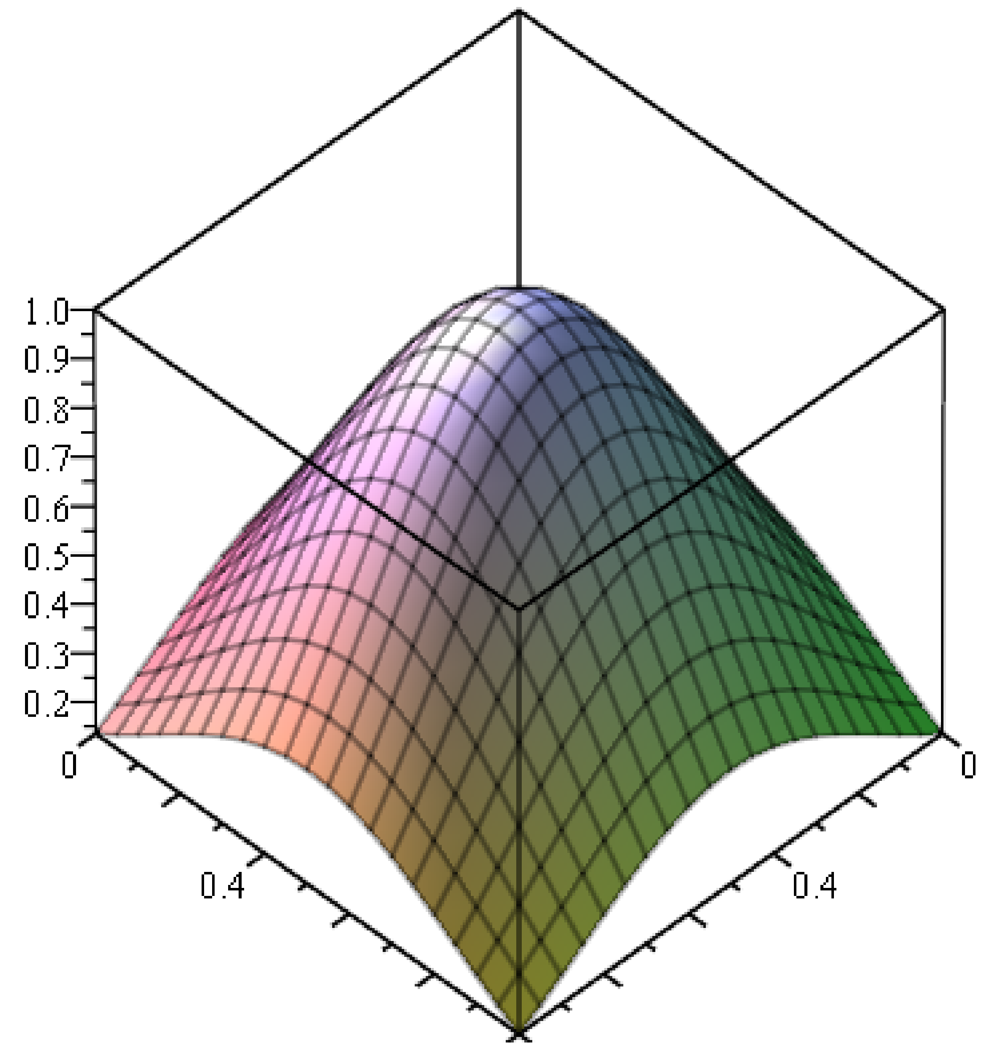
Many Different Plots with One Command

With the dataplot command, you can generate a large variety of plots by simply specifying the type of plot you want. As a shortcut, you can also select **Plots > Data Plot** from the context menu by right clicking on the following matrix in order to choose the type of data plot that you want to show.

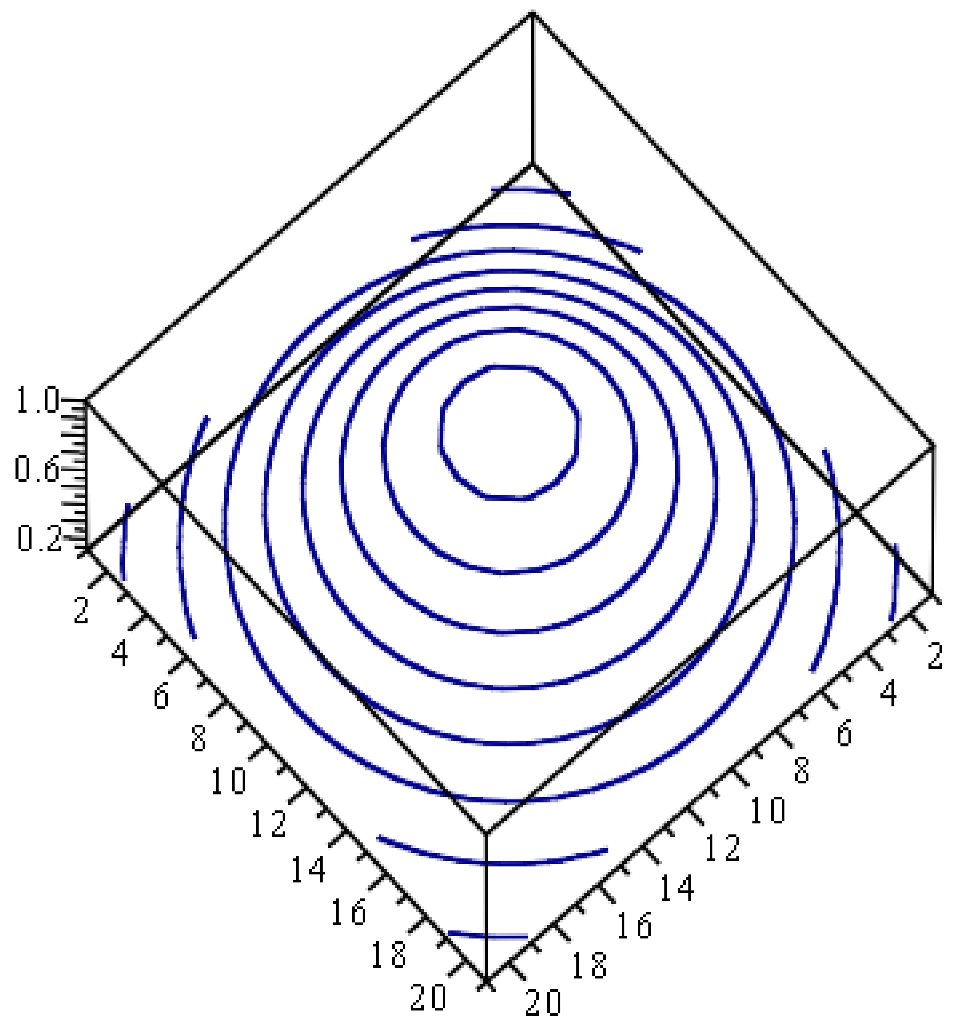
```
> M := Matrix([seq([seq(exp(-(x^2 + y^2) * (1/100)), x = -10.0 .. 10.0)], y = -10.0
    .. 10.0)], datatype = float[8])
```

$$M := \begin{bmatrix} \textit{21 x 21 Matrix} \\ \textit{Data Type: } float_8 \\ \textit{Storage: rectangular} \\ \textit{Order: Fortran\_order} \end{bmatrix} \quad (1.1)$$

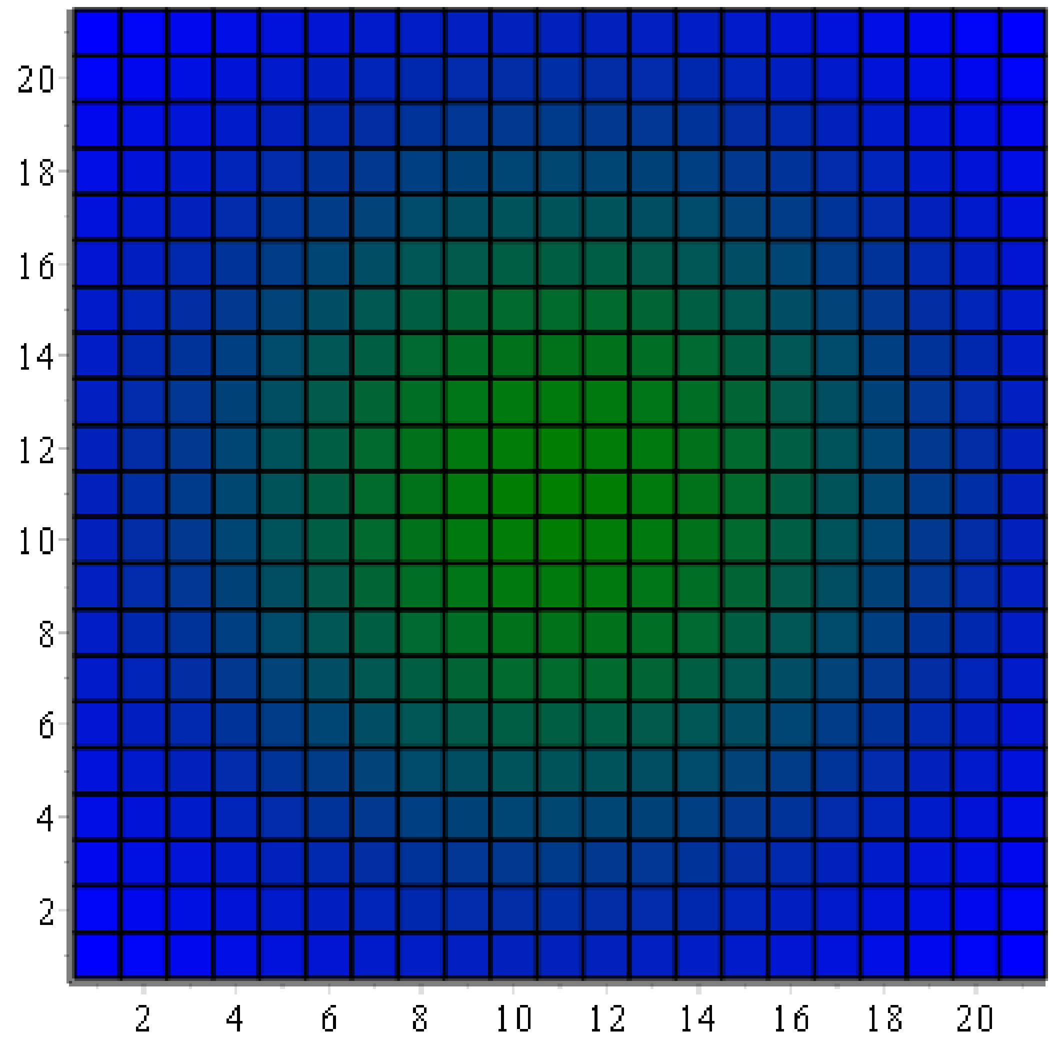
```
> dataplot(M, surface)
```



```
> dataplot(M, contour3d, color = "DarkBlue")
```



```
> dataplot(M, density, colorscheme = ["Blue", "Green"])
```



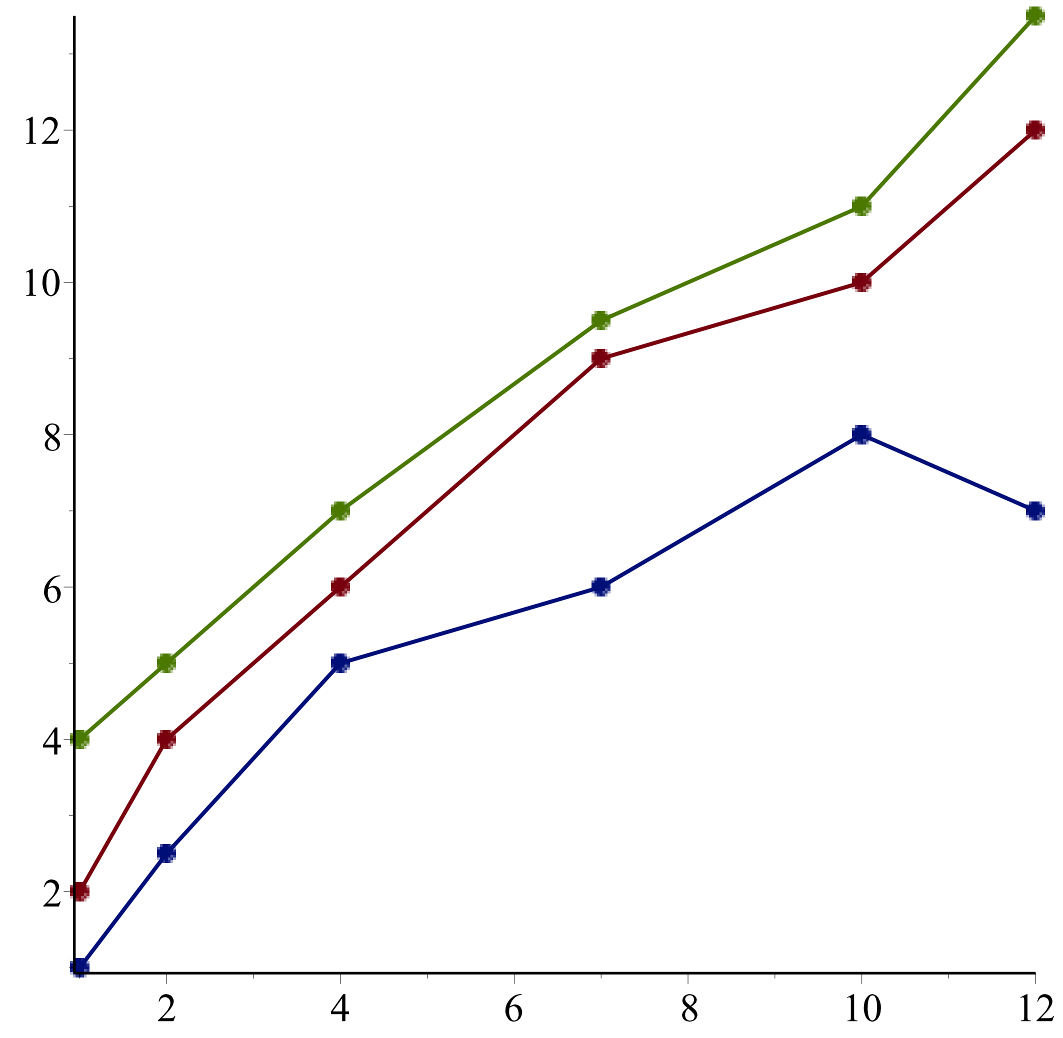
New Intuitive Calling Sequences and Support for Different Data Types

The dataplot command allows several calling sequences, making it easier to generate plots without having to transform your data into the right format. In addition to the calling sequence shown in the examples earlier, two more are available. Notice also that the data can be provided as a list, Vector, Matrix, or Array.

```
> X := Vector([1, 2, 4, 7, 10, 12], datatype=float[8]) :
  Y1 := Vector([1, 2.5, 5, 6, 8, 7], datatype=float[8]) :
  Y2 := Vector([2, 4, 6, 9, 10, 12], datatype=float[8]) :
  Y3 := Vector([4, 5, 7, 9.5, 11, 13.5], datatype=float[8]) :
```

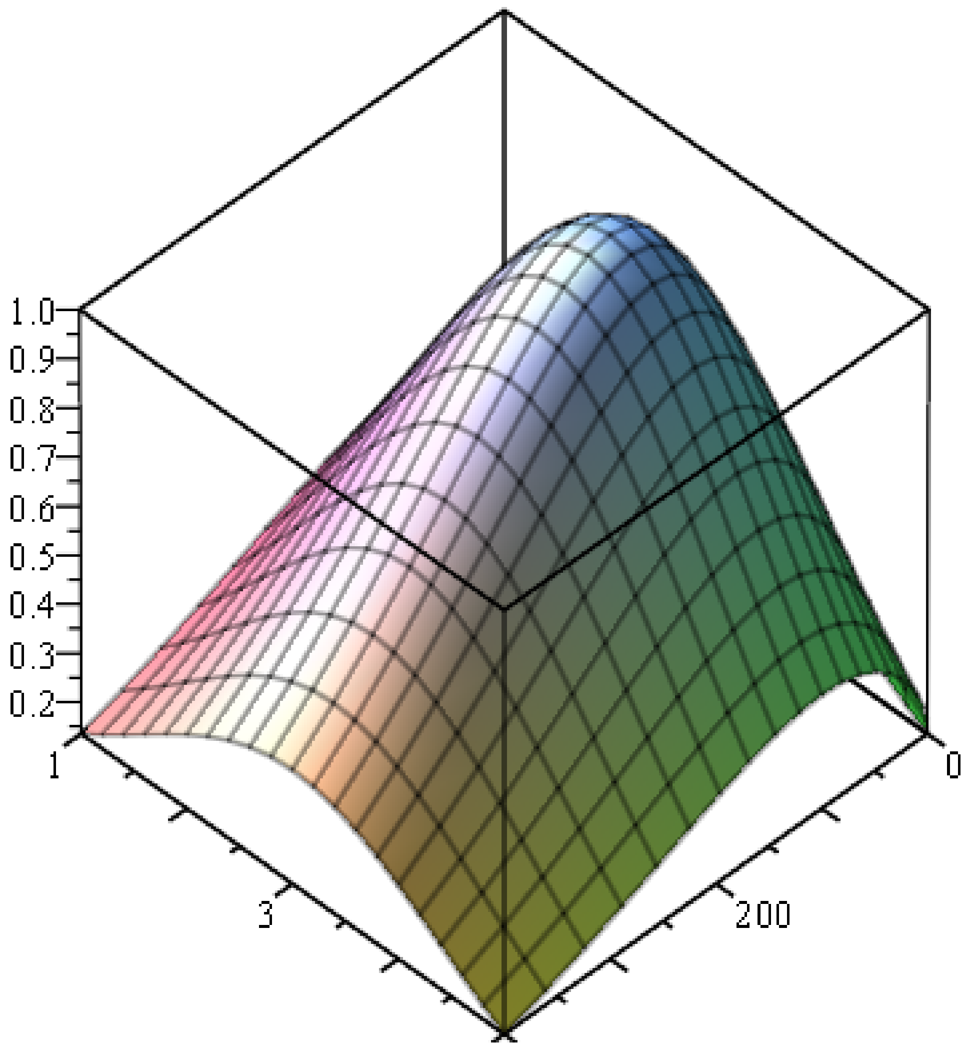
This calling sequence for 2-D point plots makes it easy to plot different sets of y-values against a single set of x-values.

```
> dataplot(X, [Y1, Y2, Y3]);
```



This calling sequence for 3-D surfaces allows you to adjust the x- and y- values associated with a grid of z-values.

```
> M := Matrix([seq([seq(exp(-(x^2 + y^2) * (1/100)), x = -10.0 .. 10.0)], y = -10.0
        .. 10.0)], datatype = float[8]) :
  dataplot([seq(i^2, i = 0 ..20)], 1 ..5, M);
```

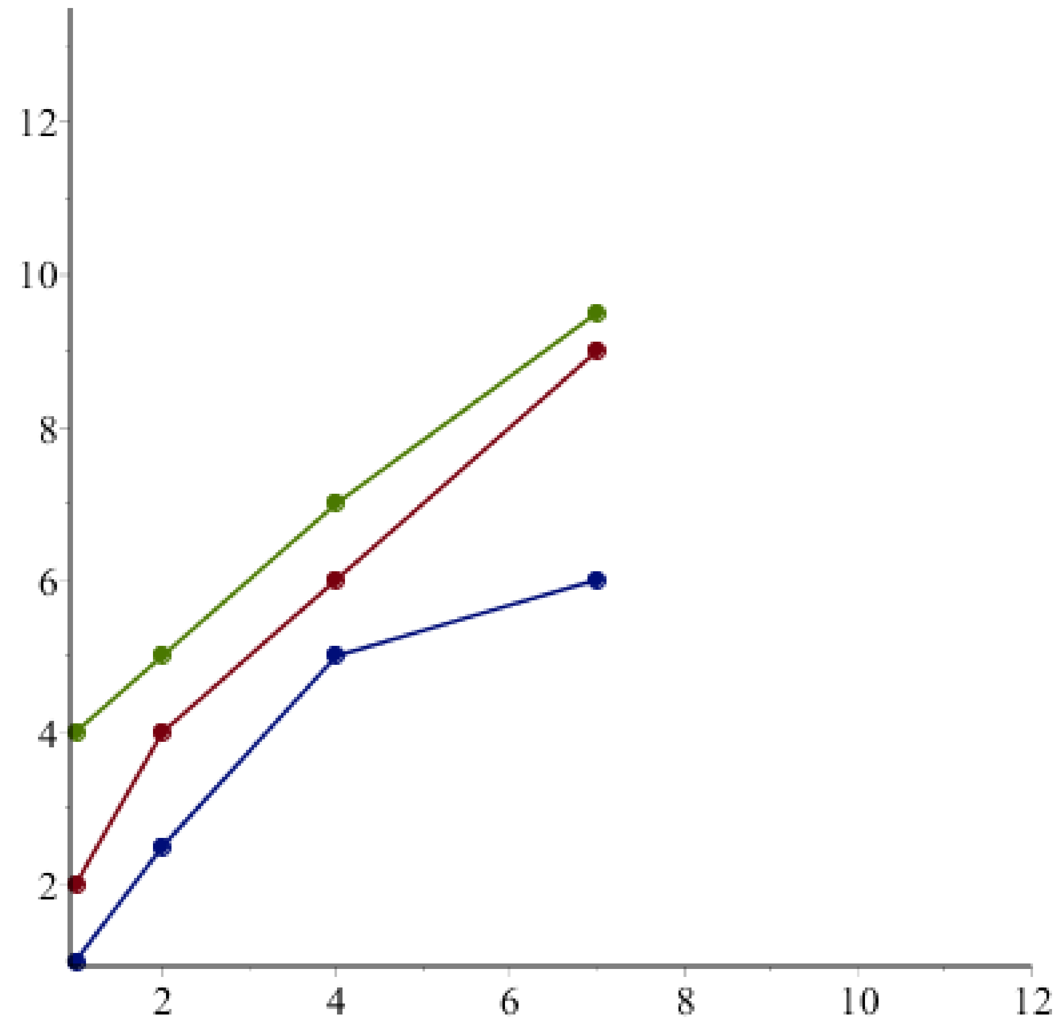


More Options for 2-D Point Plots

A number of options available with the dataplot command allow you to change the look of 2-D point plots.

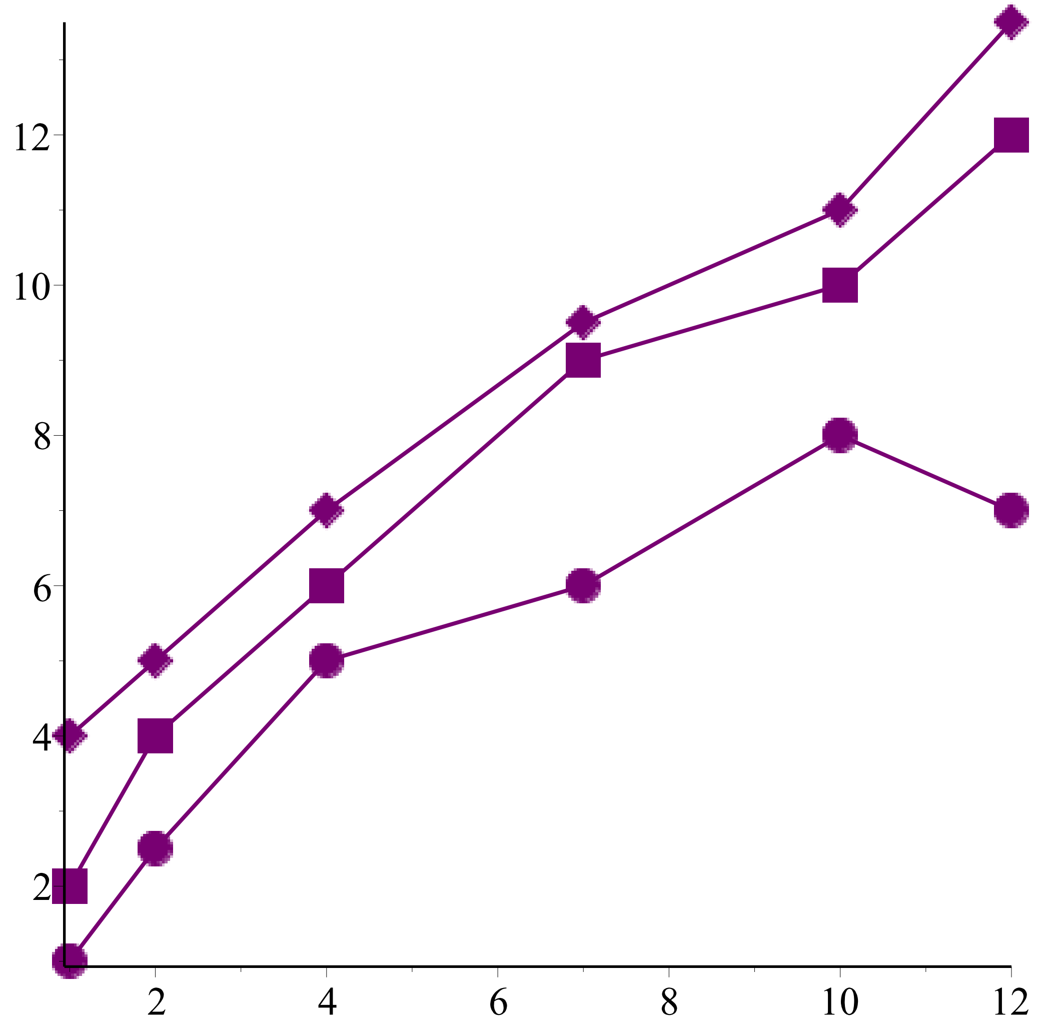
Animate a point plot. In order to view the animation, rightclick and choose - **Animation > Play**.

```
> dataplot(X, [Y1, Y2, Y3], animation)
```



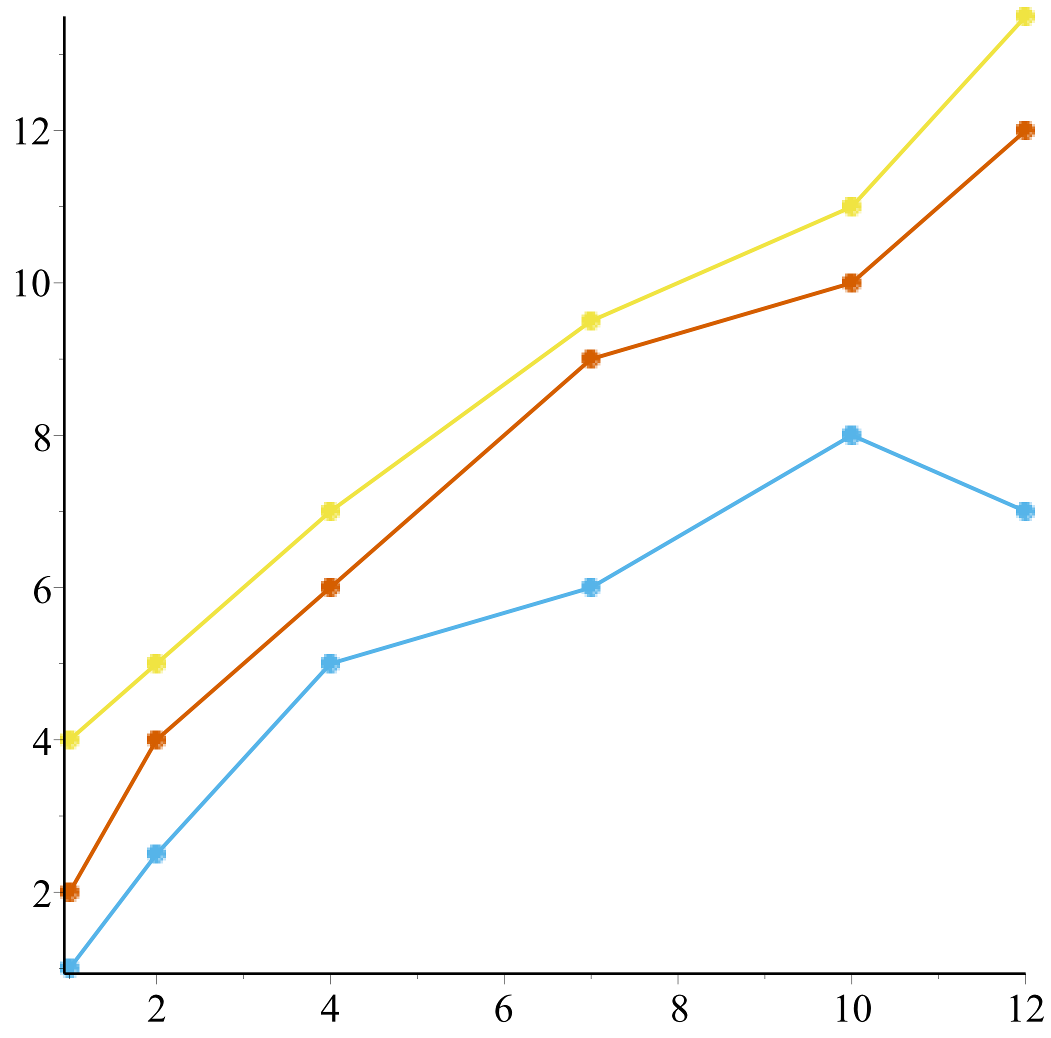
The dataplot automatically assigns colors to different datasets, but if you specify a single color, symbols are used to differentiate the datasets.

> $dataplot(X, [Y1, Y2, Y3], color = \text{"Niagara Purple"}, symbolsize = 25)$



The colorpalette option changes the color palette from which the default colors are chosen.

```
> dataplot(X, [Y1, Y2, Y3], colorpalette = "Dalton");
```

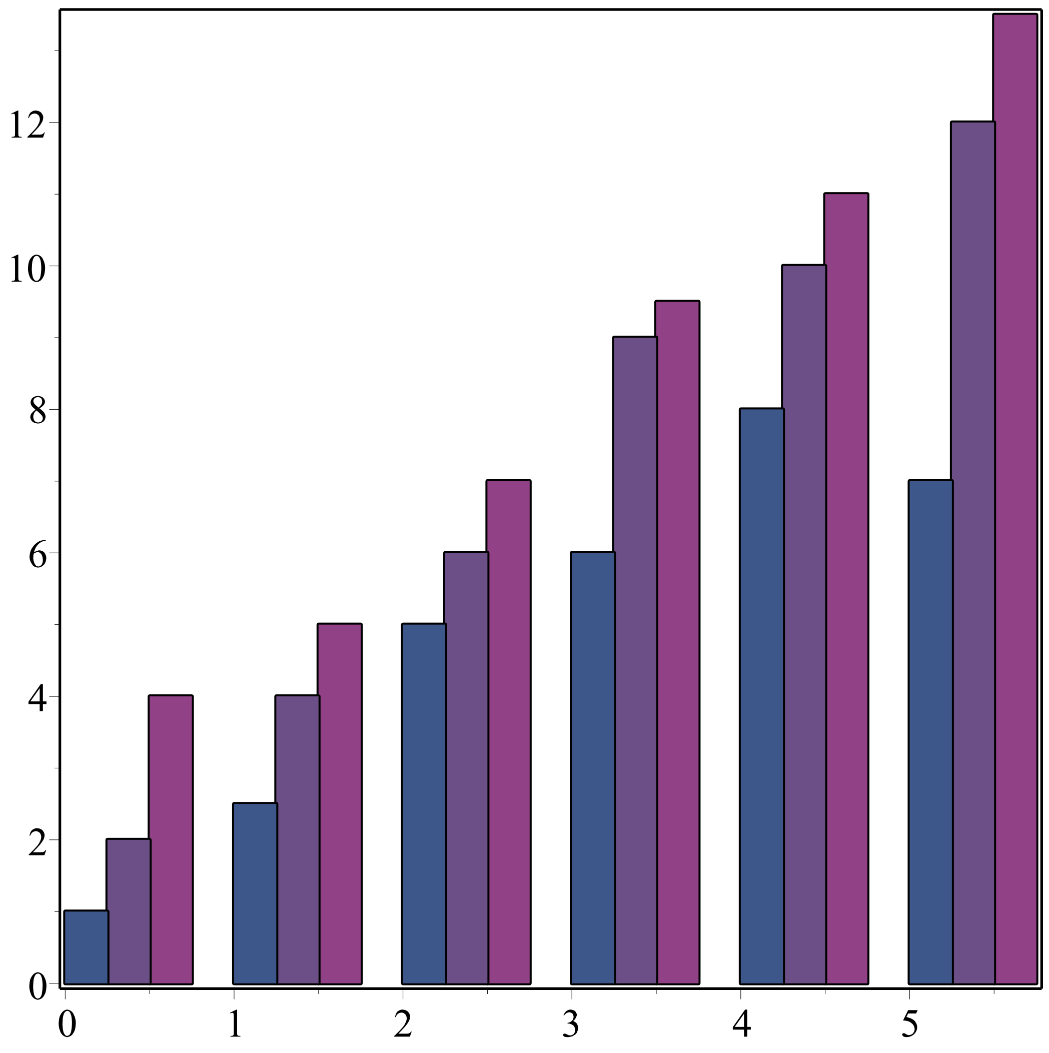


▼ Statistical Plots

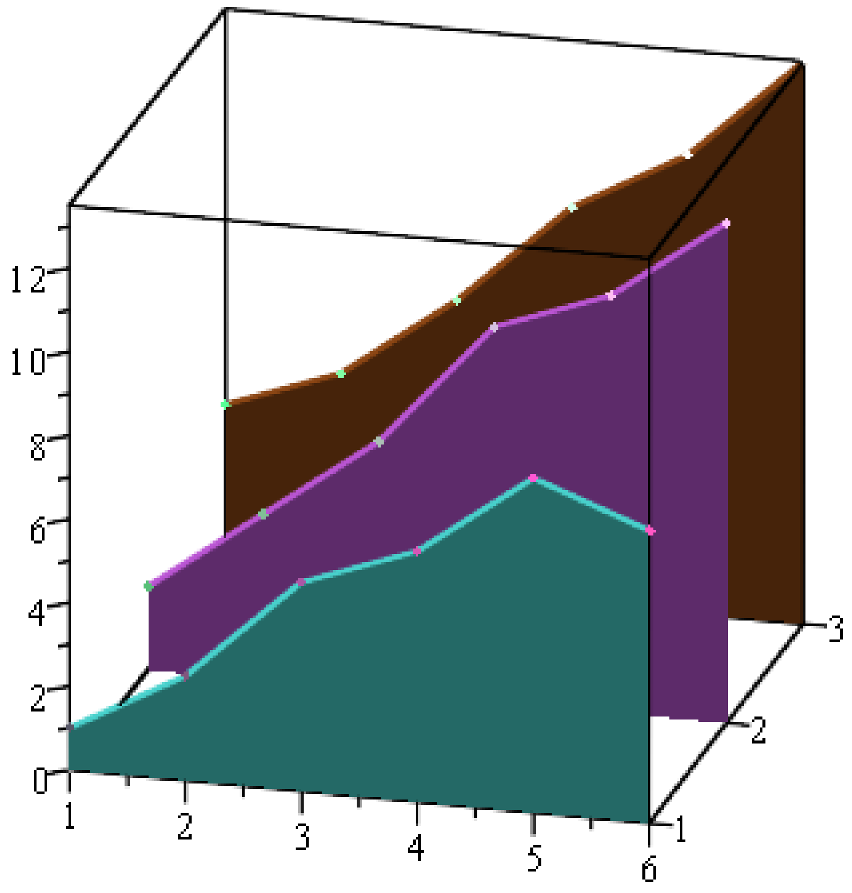
The dataplot command allows you to generate a variety of statistical plots and to visualize Quandl datasets.

Statistical plots such as bar charts and area charts are available.

```
> dataplot([Y1, Y2, Y3], bar)
```

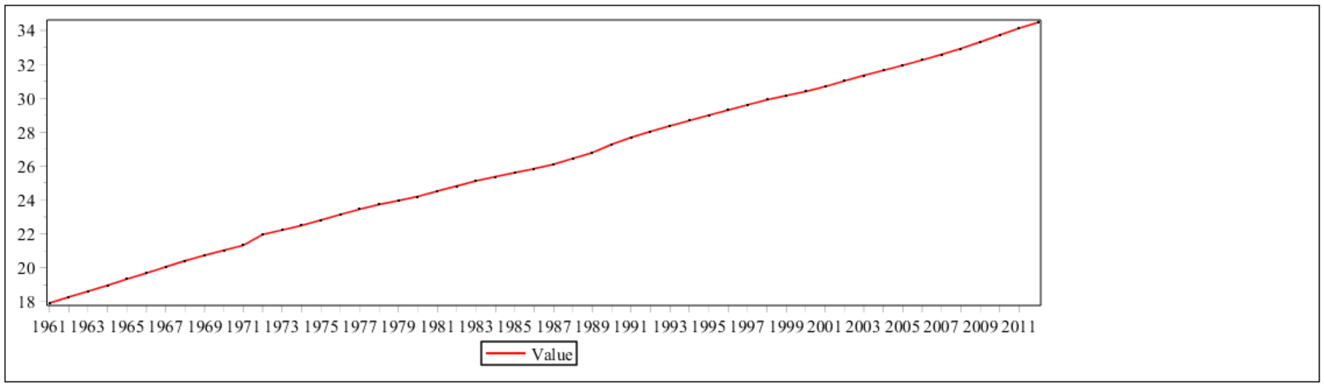


```
> dataplot([Y1, Y2, Y3], areachart, color = ["MediumTurquoise", "MediumOrchid",
        "SaddleBrown"])
```



Quandl datasets can also be plotted. The following plot shows the population in Canada:

```
> ref := DataSets:-Quandl:-Reference("FRED/CANPOPL") :
  dataplot(ref, color = "Red");
```


Value