

---

# **NMath Stats Visualization Using the Microsoft Chart Controls**



---

## Introduction

NMath Stats is part of CenterSpace Software's NMath™ product suite, which provides object-oriented components for mathematical, engineering, scientific, and financial applications on the .NET platform. NMath Stats provides functions and data structures for statistical computation, including descriptive statistics, probability distributions, combinatorial functions, multiple linear regression, hypothesis testing, and analysis of variance.

The `NMathStatsChartMicrosoft.dll` assembly included with NMath Stats provides convenience methods for plotting NMath Stats types using the Microsoft Chart Controls for .NET. This document describes how to use these methods.

**NOTE—NMath provides the NMathChart adapter class for plotting NMath types using the Microsoft Chart Controls for .NET. Use of this class, as well as an overview of the Microsoft Chart Controls data model, are presented in the CenterSpace whitepaper “NMath Visualization Using the Microsoft Chart Controls”. It is recommended that you read this document before proceeding, if you have not done so already.**

## Microsoft Chart Controls for .NET

The Microsoft Chart Controls for .NET are available as a separate download for .NET 3.5.

<http://www.microsoft.com/downloads/en/details.aspx?FamilyId=130F7986-BF49-4FE5-9CA8-910AE6EA442C>

Beginning in .NET 4.0, the Chart controls are part of the .NET Framework.

To use the Chart controls, add a reference to `System.Windows.Forms.DataVisualization` and using statements

```
using System.Windows.Forms;
using System.Windows.Forms.DataVisualization.Charting;
```

# NMath Stats Chart Adapter

**NMath Stats** provides the **NMathStatsChart** adapter class for plotting **NMath Stats** types using the Microsoft Chart Controls. **NMathStatsChart** extends **NMathChart**.

To use **NMathStatsChart**, add references to:

```
NMathChartMicrosoft.dll  
NMathStatsChartMicrosoft.dll
```

and add `using` statement

```
using CenterSpace.NMath.Charting.Microsoft;
```

The design of **NMathStatsChart** is analogous to **NMathChart**. Overloads of `ToChart()` are provided for common **NMath Stats** types. `ToChart()` returns an instance of `System.Windows.Forms.DataVisualization.Charting.Chart`. For instance:

```
DoubleMatrix data =  
    new DoubleMatrix( 30, 8, new RandGenUniform() );  
KMeansClustering km = new KMeansClustering( data );  
ClusterSet clusters = km.Cluster( 5 );  
  
xColIndex = 1;  
yColIndex = 3;  
Chart chart =  
    NMathStatsChart.ToChart( clusters, data, xColIndex, yColIndex );
```

The returned chart can be customized as desired.

```
chart.Titles.Add("Hello World");
```

Similarly:

- The default look of generated charts is governed by static properties on **NMathChart** class: `DefaultSize`, `DefaultTitleFont`, `DefaultAxisTitleFont`, `DefaultMajorGridLineColor`, and `DefaultMarker`.
- For prototyping and debugging, the `Show()` function shows a given chart in a default form. The chart is disposed when the window is closed.
- The `Update()` function is used to update a chart control added in the Designer with an **NMath Stats** object.

- The `Save()` function saves a chart to a file or stream.
- Class `NMathChart.Unit` represents a unit of physical quantity, and can be used to specify axis units in cases where the `NMath Stats` object does not provide this information.

For more information on any of these topics, see “NMath Visualization Using the Microsoft Chart Controls”.

## Plotting Data Frames

`NMath Stats` provides the `DataFrame` class which represents a two-dimensional data object consisting of a list of columns of the same length. Columns are themselves lists of different types of data: numeric, string, boolean, generic, and so on. `NMath Stats` provides two numeric column types: `DFIntColumn` represents a column of integer data, and `DFNumericColumn` represents a column of double-precision floating point data.

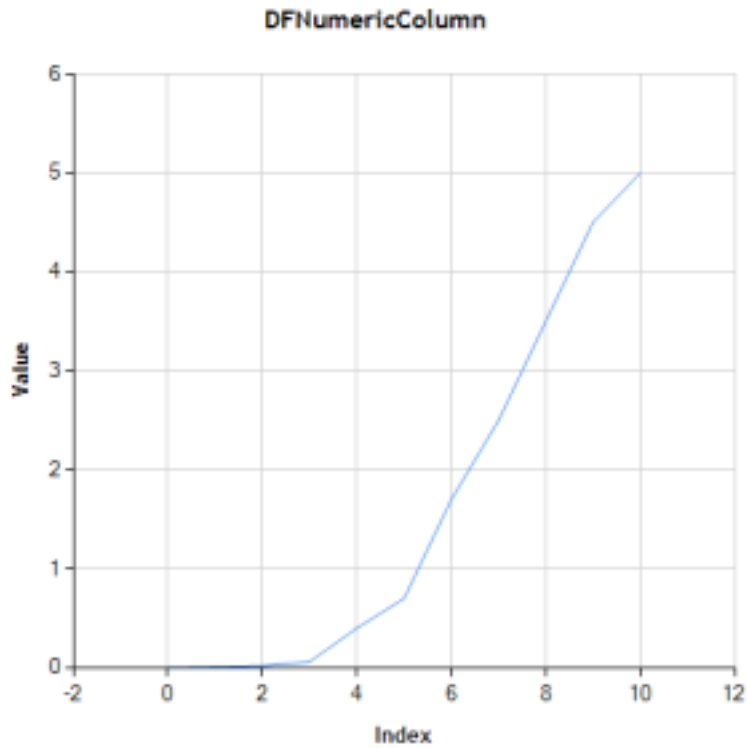
`NMathStatsChart` plots numeric data held in a `DataFrame` or individual numeric columns. The plotting options for data frames are analogous to the plotting options for vectors and matrices.

**NOTE—When plotting full data frames, non-numeric columns are ignored.**

For example, a single numeric column can be plotted as line data series, with the  $x$ -values taken from the column indices, or the  $x$ -values can be taken from a specified `NMathChart.Unit` object.

```
DFNumericColumn col = new DFNumericColumn( "MyColumn",
    new double[] { 0.0, 0.01, 0.02, 0.06, 0.4, 0.7, 1.7, 2.5, 3.5,
                  4.5, 5.0 } );
NMathStatsChart.Show( col );
```

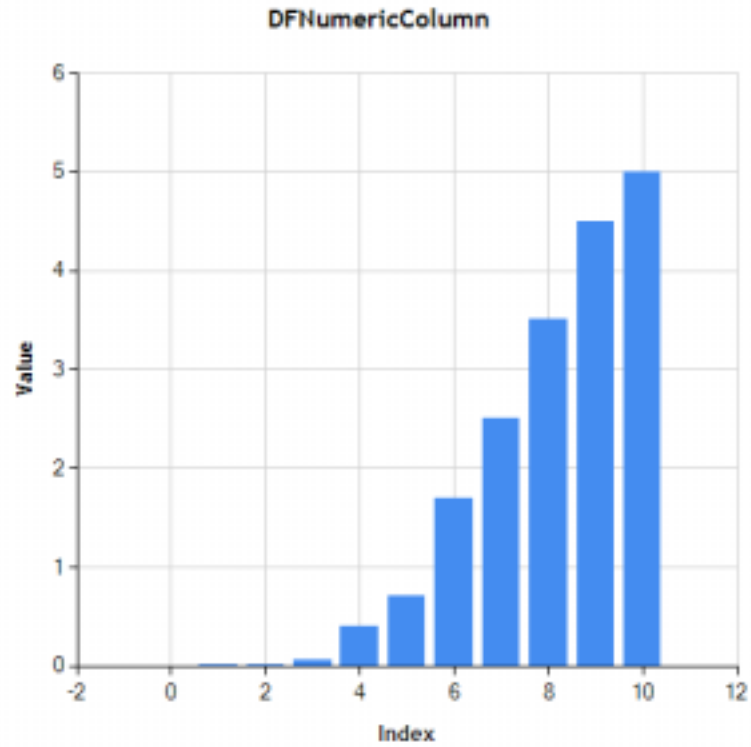
Figure 1 – Numeric column series



The generated chart uses a line chart to display the data series, but this can easily be customized. For instance, this code uses a column chart.

```
Chart chart = NMathStatsChart.ToChart( col );  
chart.Series[0].ChartType = SeriesChartType.Column;  
NMathChart.Show( chart );
```

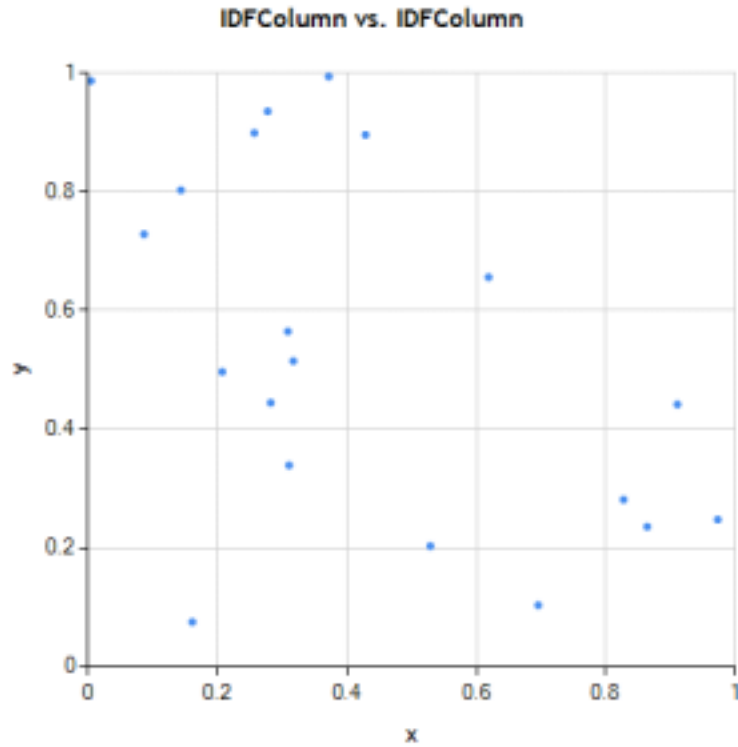
Figure 2 – Column chart



Two columns can also be plotted against one another in a scatter plot. For example:

```
DoubleMatrix A = new DoubleMatrix( 20, 3, new RandGenUniform() );  
DataFrame df =  
    new DataFrame( A, new string[] { "col1", "col2", "col3" } );  
NMathStatsChart.Show( df["col1"], df["col3"] );
```

Figure 3 – Scatter plot

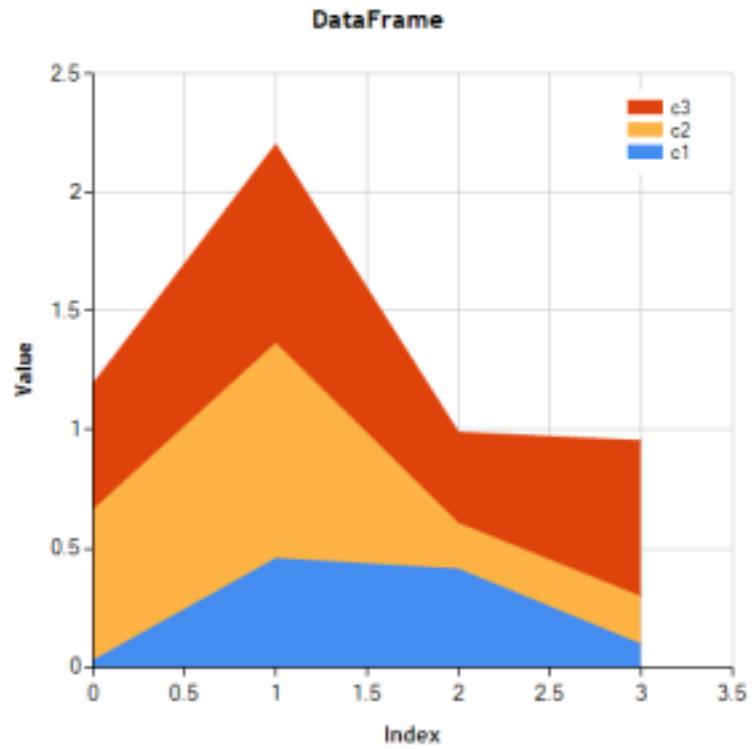


A full **DataFrame** can also be plotted. By default, each numeric column is plotted as a separate data series. (Non-numeric columns are ignored.) The generated chart uses a line chart to display each series, but this can easily be customized. For instance, this code uses a stacked area chart.

```
DoubleMatrix A = new DoubleMatrix( 4, 3, new RandGenUniform() );
DataFrame df =
    new DataFrame( A, new string[] { "c1", "c2", "c3" } );
df.AddColumn( new DFStringColumn( "c4", new string[] { "once",
    "upon", "a", "time" } ) );
Chart chart = NMathStatsChart.ToChart( df );

foreach( Series series in chart.Series )
{
    series.ChartType = SeriesChartType.StackedArea;
}
chart.ChartAreas[0].AxisX.Minimum = 0;
NMathStatsChart.Show( chart );
```

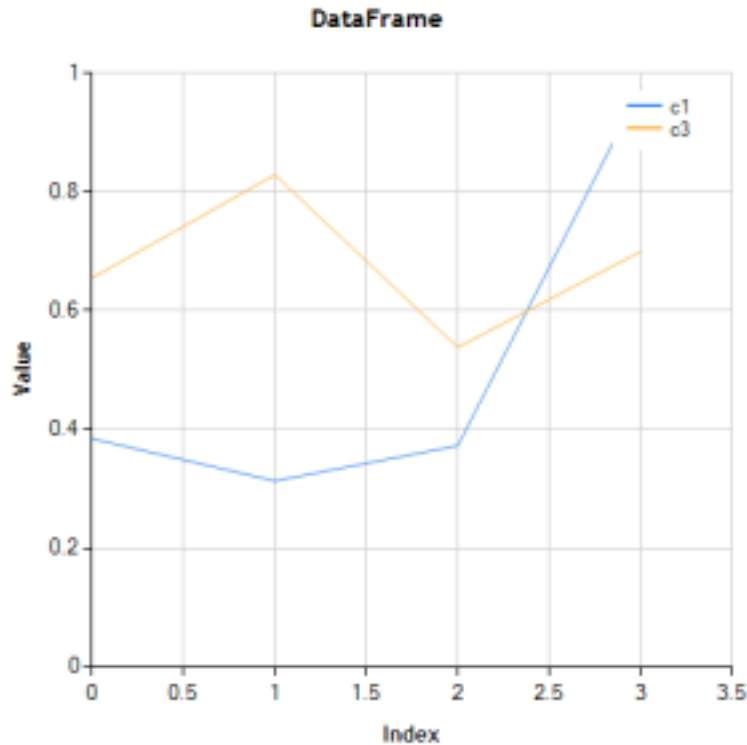
Figure 4 – Data frame columns with custom chart type



By default, all numeric columns are plotted, but you can optionally specify an array of column indices to plot.

```
int[] colIndices = new int[] { 0, 2 };  
Chart chart = NMathStatsChart.ToChart( df, colIndices );  
  
chart.ChartAreas[0].AxisX.Minimum = 0;  
NMathStatsChart.Show( chart );
```

Figure 5 – Data frame columns by index



## Plotting Probability Distributions

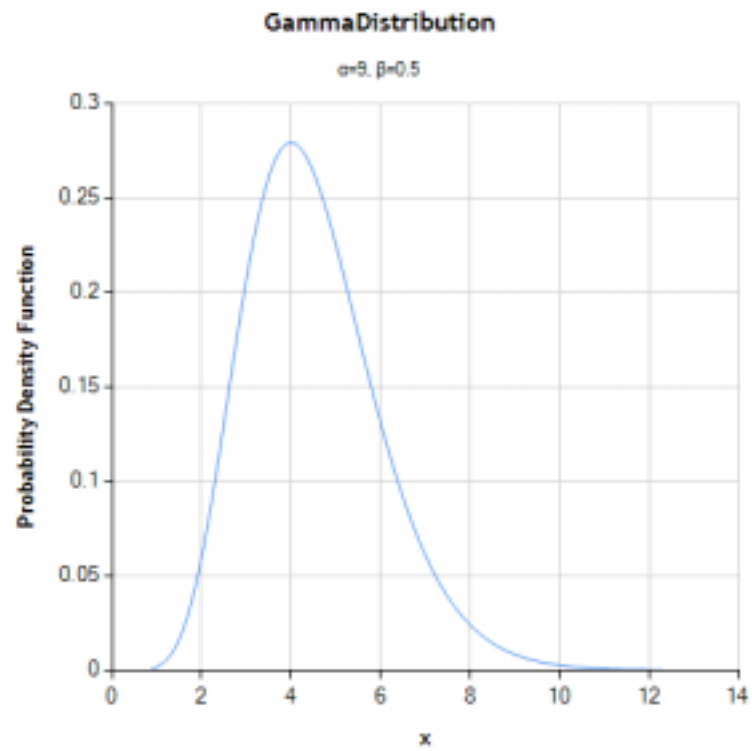
**NMath Stats** provides classes for computing the probability density function (PDF) and the cumulative distribution function (CDF) for a variety of probability distributions, including beta, binomial, chi-square, exponential, F, gamma, geometric, Johnson, logistic, log-normal, negative binomial, normal (Gaussian), Poisson, Student's t, triangular, uniform, and Weibull distributions.

**NMathStatsChart** plots continuous **NMath Stats** probability distributions by interpolating over the PDF or CDF function. The function to plot is specified using a value from the `NMathStatsChart.DistributionFunction` enumeration. For instance:

```
double alpha = 9.0;  
double beta = 0.5;  
GammaDistribution gamma = new GammaDistribution( alpha, beta );
```

```
NMathStatsChart.Show( gamma,  
    NMathStatsChart.DistributionFunction.PDF );
```

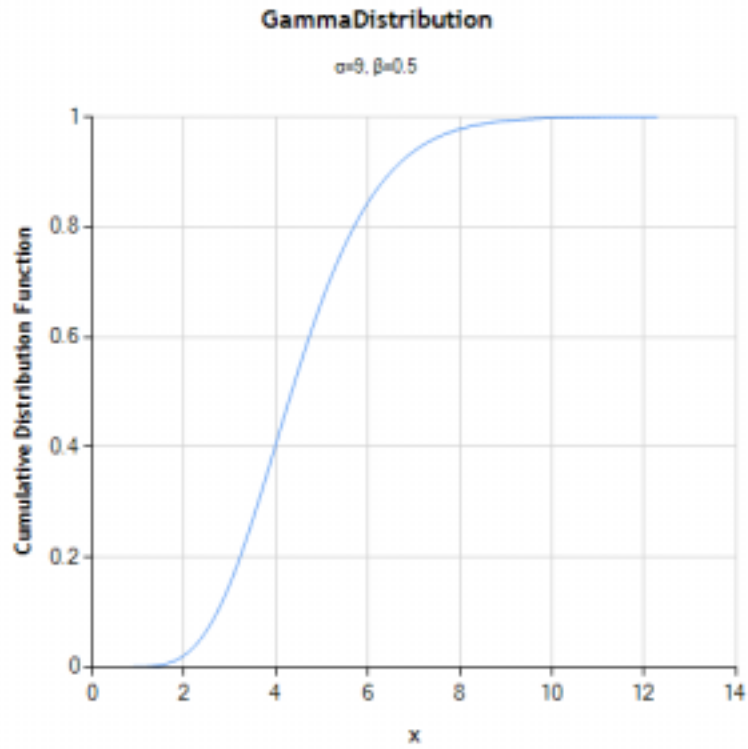
Figure 6 – Gamma distribution PDF



The distribution parameters are shown in a subtitles. Similarly:

```
NMathStatsChart.Show( gamma,  
    NMathStatsChart.DistributionFunction.CDF );
```

Figure 7 – Gamma distribution CDF

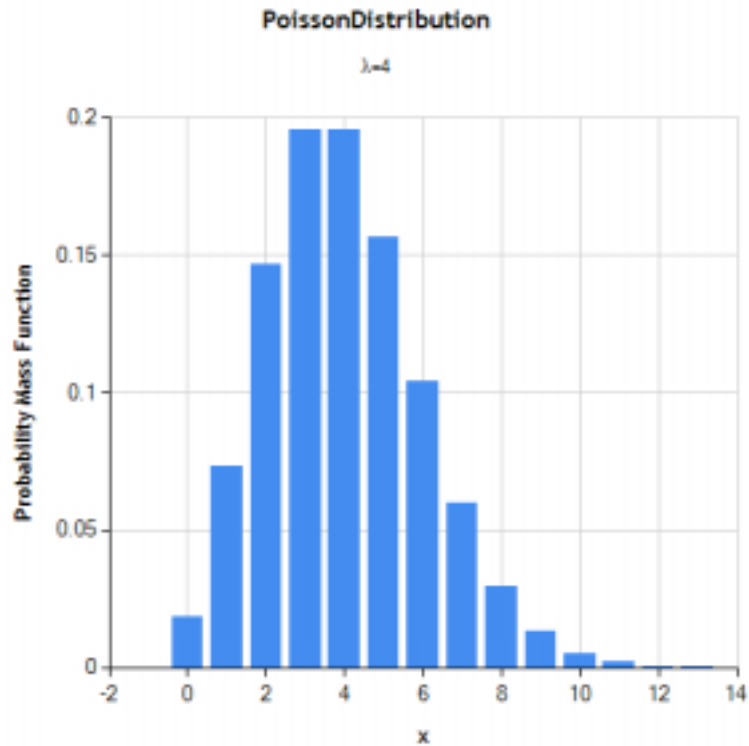


For discrete distributions, a column chart is used.

```
double lambda = 4.0;
PoissonDistribution poisson = new PoissonDistribution( lambda );

NMathStatsChart.Show( poisson,
    NMathStatsChart.DistributionFunction.PDF );
```

Figure 8 – Poisson distribution PDF



## Plotting Linear Regressions

In **NMath Stats**, class **LinearRegression** computes a multiple linear regression from an input matrix of independent variable values (the *predictor* matrix or *regression* matrix) and a vector of dependent variable values (the *observation* vector).

**NMathStatsChart** plots linear regressions. You must also specify the predictor (independent) variable to plot on the  $x$ -axis. For example:

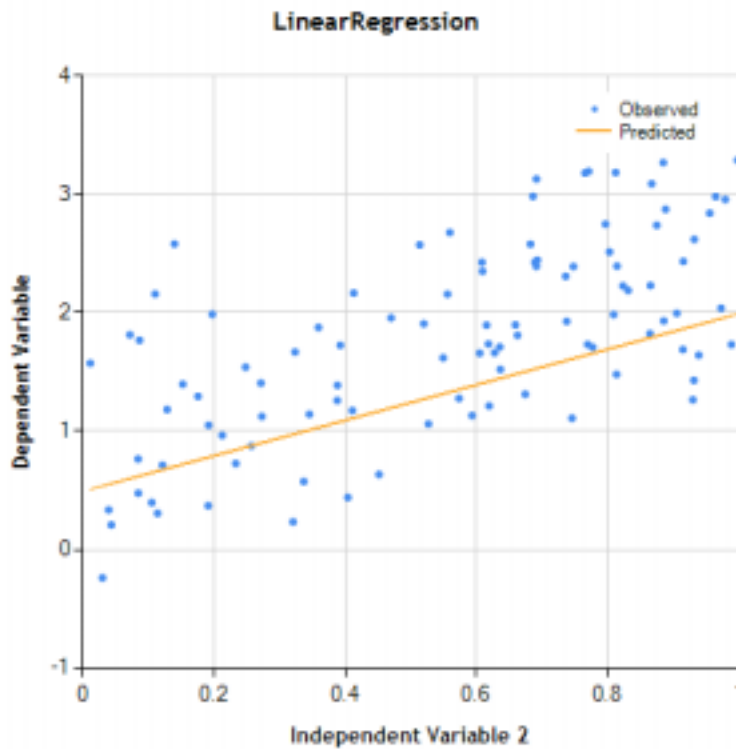
```
DoubleMatrix predictors =  
    new DoubleMatrix( 100, 3, new RandGenUniform() ); ;  
DoubleVector observations = 2 * predictors.Col(0) +  
    -0.75 * predictors.Col(1) + 1.25 * predictors.Col(2) + 0.5;  
bool addIntercept = true;
```

```

LinearRegression lr =
    new LinearRegression( predictors, observations, addIntercept );
NMathStatsChart.Show( lr, 2 );

```

Figure 9 – Linear regression



Note that the multidimensional linear regression fit is plotted projected onto the plane of the specified predictor variable. In this case, although the model has three independent variables, we are plotting only the third independent variable versus the dependent variable for the fitted model (plus intercept, if the model has an intercept parameter). The full model is:

$$y = 2 * x_0 - x_1 + 1.5 * x_2 + 0.5$$

Because predictor index 2 is specified, the plotted line is

$$y = 1.5 * x_2 + 0.5$$

## Plotting Goodness of Fit

**NMath Stats** provides class **GoodnessOfFit** for testing the goodness of fit of least squares model-fitting classes, such as **LinearRegression**, **PolynomialLeastSquares**, and **OneVariableFunctionFitter**:

Available statistics include the residual standard error, the coefficient of determination ( $R^2$  and “adjusted”  $R^2$ ), the F-statistic for the overall model with its numerator and denominator degrees of freedom, as well as statistics for the individual model parameters.

**NMathStatsChart** plots **GoodnessOfFit** objects. Overall model statistics are shown in subtitles, and individual parameter values and confidence intervals are shown in a column chart with error bars.

```
DoubleVector x = new DoubleVector( 10, .01, 0.1 );
DoubleVector y = new DoubleVector( 0.06, 0.27, 0.51, 0.73, 0.92,
    1.1, 1.16, 1.31, 1.42, 1.52 );

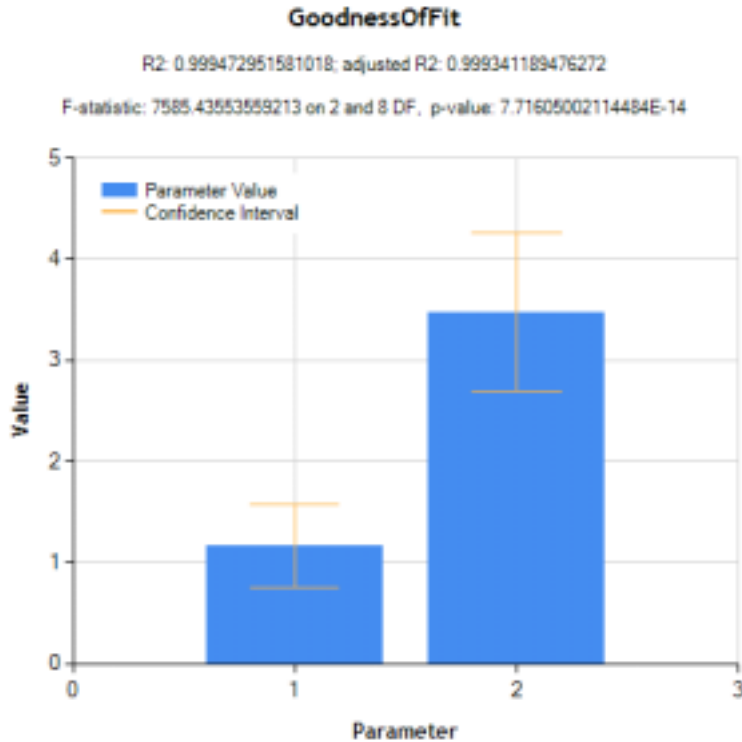
Func<DoubleVector, double, double> f = delegate( DoubleVector p,
double xval )
{
    double K = p[0];
    double Vm = p[1];
    return Vm * xval / ( K + xval );
};

DoubleVector start = new DoubleVector( "0.1 0.1" );
OneVariableFunctionFitter<TrustRegionMinimizer> fitter =
    new OneVariableFunctionFitter<TrustRegionMinimizer>( f );

DoubleVector solution = fitter.Fit( x, y, start );
GoodnessOfFit gof = new GoodnessOfFit( fitter, x, y, solution );

double alpha = 0.01;
Chart chart = NMathStatsChart.ToChart( gof, alpha );
chart.Legends[0].Docking = Docking.Left;
NMathStatsChart.Show( chart );
```

Figure 10 – Goodness of fit



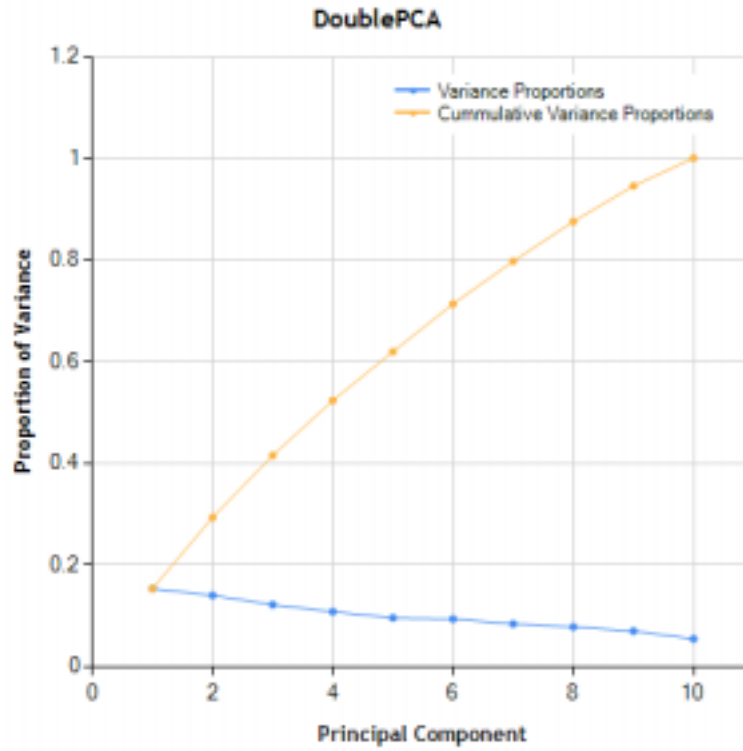
## Plotting Principal Component Analysis

Principal component analysis (PCA) finds a smaller set of synthetic variables that capture the variance in an original data set. The first principal component accounts for as much of the variability in the data as possible, and each succeeding orthogonal component accounts for as much of the remaining variability as possible. In **NMath Stats**, classes **DoublePCA** and **FloatPCA** perform principal component analyses.

**NMathStatsChart** plots principal component analyses. By default, a Scree plot is generated, showing the fraction of total variance in the data captured by each principal component.

```
DoubleMatrix A = new DoubleMatrix( 100, 10, new RandGenUniform() );  
DoublePCA pca = new DoublePCA( A );  
  
NMathStatsChart.Show( pca );
```

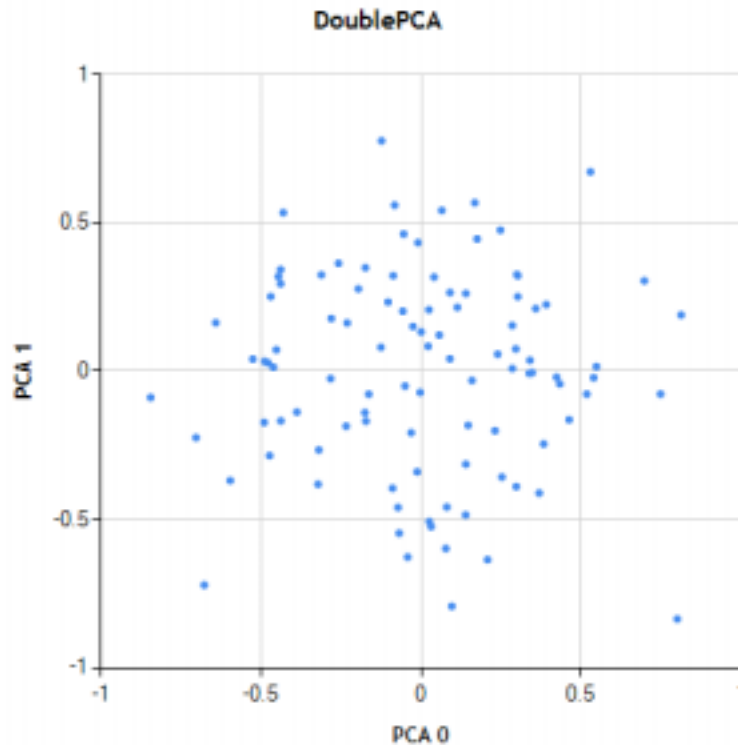
Figure 11 – Scree plot



Alternatively, you can plot the original data in the plane of two specified principal components. For instance:

```
int xIndex = 0;  
int yIndex = 1;  
NMathStatsChart.Show( pca, xIndex, yIndex );
```

Figure 12 – Plane of first two principal components



## Plotting Cluster Analyses

Cluster analysis detects natural groupings in data. **NMath Stats** provides several classes for performing cluster analysis:

- Class **ClusterAnalysis** performs hierarchical cluster analysis. In hierarchical cluster analysis, each object is initially assigned to its own singleton cluster. The analysis then proceeds iteratively, at each stage joining the two most similar clusters into a new cluster, continuing until there is one overall cluster.
- Class **KMeansClustering** performs  $k$ -means clustering. The  $k$ -means clustering method assigns data points into  $k$  groups such that the sum of squares from points to the computed cluster centers is minimized.
- Class **NMFClustering** performs clustering using iterative nonnegative matrix factorization (NMF)

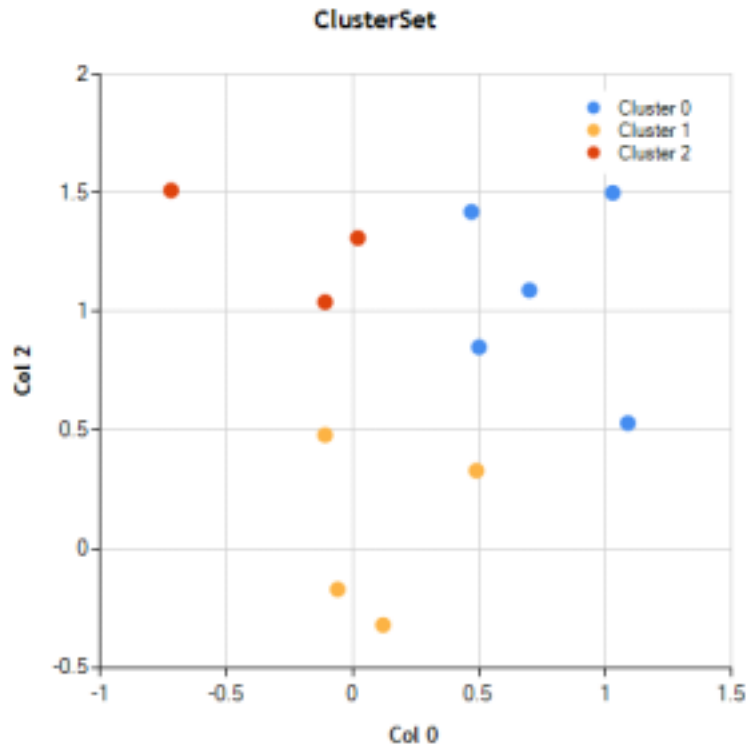
All cluster analysis classes encapsulate cluster results using instances of **ClusterSet**, which describes a collection of objects assigned to a finite number of clusters. **NMathStatsChart** plots **ClusterSet** instances using a scatter plot, assigning points different colors based on the cluster assignment. For example:

```
DoubleMatrix data = new DoubleMatrix( "12x3 [ 0.49 0.18 0.33
                                             0.12 0.83 -0.32
                                             -0.11 0.28 0.48
                                             -0.06 -0.28 -0.17
                                             0.50 0.81 0.85
                                             0.70 0.22 1.09
                                             1.09 1.27 0.53
                                             1.03 0.52 1.50
                                             0.02 0.58 1.31
                                             -0.72 0.24 1.51
                                             0.47 0.33 1.42
                                             -0.11 0.45 1.04 ]" );

KMeansClustering km = new KMeansClustering( data );
ClusterSet clusters = km.Cluster( 3 );

int xColIndex = 0;
int yColIndex = 2;
chart =
    NMathStatsChart.ToChart( clusters, data, xColIndex, yColIndex );
foreach (Series series in chart.Series)
{
    series.MarkerSize = 10;
}
NMathStatsChart.Show( chart );
```

Figure 13 – Cluster set



## Conclusions

NMath Stats types can be easily plotted using the NMathStatsChart adapter and the free Microsoft Chart Controls for .NET, creating a complete solution for statistical data analysis and visualization.

## **NMATH STATS VISUALIZATION USING THE MICROSOFT CHART CONTROLS**

© 2011 Copyright CenterSpace Software, LLC. All Rights Reserved.

The correct bibliographic reference for this document is:

*NMath Stats Visualization Using the Microsoft Chart Controls*, CenterSpace Software, Corvallis, OR.

Printed in the United States.

Printing Date: August, 2011

### **CENTERSPACE SOFTWARE**

Address:	230 SW 3rd St., Suite 311, Corvallis, OR 97333 USA
Phone:	(541) 896-1301
Web:	<a href="http://www.centerspace.net">http://www.centerspace.net</a>
Technical Support:	<a href="mailto:support@centerspace.net">support@centerspace.net</a>