Transforms and Regressions

Chapter 1

Introduction

Chapter 2

Transforms, Regressions, and Graph Analysis

Topics:

This document describes all of SigmaPlot's powerful math features. These can be found in the **Transform**, **Nonlinear Regression**, and **Graph Analysis** groups on the **Analysis** tab.These can be found in the **Transform** and **Graph Analysis** groups on the **Analysis** tab.

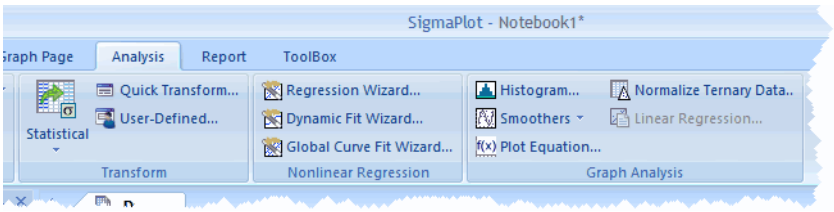


Figure 1: Transform, Nonlinear Regression, and Graph Analysis groups on the Analysis tab

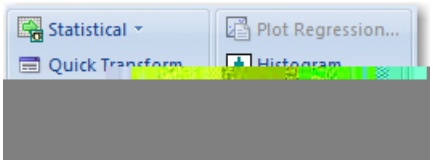


Figure 2: Transform and Graph Analysis groups on the Analysis tab

Transforms are sets of equations that manipulate and calculate data. Math transforms apply math functions to existing data and also generate serial and random data. To perform a transform, you enter variables and standard arithmetic and logic operators into a transform dialog box. Your equations can specify that a transform access data from a worksheet as well as save equation results to a worksheet.

You can save transforms as independent .XFM files for later opening or modification. Because transforms are saved as plain text (ASCII) files, you can create and edit them using any word processor that can edit and save text files.

Statistical Transforms

SigmaPlot now includes a complete array of general data transformations.

Viewing the Statistical Transforms

To view these statistical transforms:

On the **Analysis** tab, in the **Transform** group, click the **Statistical** drop-down arrow.

General data transformations are math functions and equations which are applied to worksheet data which you can use these to transform data to better fit assumptions of tests, or otherwise modify it before performing a statistical procedure. You can use data transforms to:

- Generate random numbers.
- Define dummy variables, lagged variables, and variable interactions.
- Convert other missing values codes to the "--" double dash symbol for missing values used by SigmaStat.

These commands all appear in the **Transforms** group. Choosing these commands prompts you to select the columns to transform, followed by a result column.

Stacking Data

You can merge the contents of two or more columns by stacking the column contents on top of each other.

1. On the **Analysis** tab, in the **Transform** group, select:

   Statistical > **Stack**

   The **Stack Columns - Select Data** dialog box appears.
2. Select the output column to place the stacked data by clicking the worksheet column.
3. Select the columns to stack, either by clicking the worksheet columns, or selecting the column from the **Data for Input** drop-down list.
4. Click **Finish** to stack the contents of the selected input columns in the selected output column.

   Note that you cannot stack blocks of data, only entire columns.

Indexing Data

You can convert raw data to indexed data with one and two factors, and vice versa.

To create indexed data:

1. On the **Analysis** tab, in the **Transform** group, select:

   Statistical > **Index** > **One Way or Two Way**

   Tip: Select **One Way** to index by one factor, or **Two Way** to index for two factors.
2. In the **Select Data** dialog box that appears, select the output column to place the indexed data by clicking the worksheet column. This should be an empty column with at least one empty column to the right for a One Way ANOVA, or two empty columns for Two Way ANOVA.
3. Select the columns to index, either by clicking the worksheet columns, or selecting the column from the **Data for Input** drop-down list.
4. Click **Finish** to index the contents of the selected input columns in the selected output column.

   The indexed data is tabulated, with the indexes appearing in the left column(s), and the data in the right column.

Unindexing Data

To unindex data for graphing purposes:

1. On the **Analysis** tab, in the **Transform** group, select:
   Statistical > **Unindex** > **One Way or Two Way**

   Tip: Select **One Way** to unindex by one factor, or **Two Way** to unindex for two factors.
2. Select the columns to unindex as prompted.
3. If you unindexed two ways, each column contains the data for one cell in the Two Way ANOVA table, and the two factor levels appear as the column title, separated by a hyphen (-).

Simple Transforms

SigmaPlot provides several commonly used transformations that are used to linearize or normalize observations or stabilize the variance, particularly in regression and analysis of variance problems.

- Subtract
- Divide
- Square
- Absolute Value
- Natural log ln(x)
- Log log(x)
- Reciprocal
- Exponential
- Square Root
- Arcsin Square Root Transform

Using Simple Transforms

1. On the **Analysis** tab, in the **Transform** group, select:
   Statistical > **Simple Transforms** > **[Desired Transform]**

   The **Select Data** dialog box for the specified transform appears and prompts you for an input column.

   Note: If you select a column in the worksheet before you choose the transform, the selected column is automatically assigned as the input column, and you are prompted for the output column.
2. Pick the data column you want to apply the transform to as the input column by clicking it in the worksheet or selecting it from the **Data for Input:** drop-down list.
   The number or title of the selected column appear in the highlighted input row in the **Selected Columns** list, and you are prompted for an output column.
3. Pick the column where you want the transform results to appear as the output column by clicking it in the worksheet or selecting it form the drop-down list. The number or title of the selected column appears in the highlighted output row.
4. Click **Finish** to run the transform on the specified input column.

Centering Data

The center transform subtracts the mean of a column from all values in that column and places the result in a specified output column. You can often use the center transform on data to eliminate or reduce multicollinearity. For more information on centering data, you can reference any appropriate statistics reference.

To center a variable:

1. On the **Analysis** tab, in the **Transform** group, select:
   Statistical > **Center**

   The **Center Transform - Select Data** dialog box appears and prompts you to select an input column.

   Note: If you select a column in the worksheet before you choose the transform, the selected column is automatically assigned as the input column, and you are prompted for the output column.

2. Pick the worksheet column with the data you want to center as the input column by clicking it in the worksheet or selecting it from the **Data for Input** drop-down list. The number or title of the selected column appear in the highlighted input row and you are prompted for an output column
3. Pick the column where you want the centered variables to appear as the output column by clicking it the worksheet or selecting it from the **Data for Output** drop-down list. The number or title of the selected column appear in the highlighted output row.
4. **To change your selections,** select the column assignment in the **Selected Columns** list, then select the desired column from the worksheet or the drop-down list. You can also double-click a column assignment to clear it.
5. Click **Finish** to run the transform on the specified input column.

Note: If you specify an output column that contains data, a dialog box appears asking you if you want to erase the column contents, push the contents down, or cancel the transform. Click **Overwrite** to replace the existing column contents with the transform results. Select **Insert** to place transform results above the existing cell contents. The data from the input columns are factored together and placed in the specified output column.

Standardizing Data

Use this transform if you want to standardize variables before performing a statistical procedure. By definition, standardized data has a mean of zero and a standard deviation of one. The standardize transform subtracts the mean of a column from all values in that column, then divides the centered values by the standard deviations.

To standardize a variable:

1. On the **Analysis** tab, in the **Transform** group, select:

   Statistical > **Standardize**

   The **Standardize Transform - Select Data** dialog box appears prompting you to select an Input: column.

   Tip: If you select a column in the worksheet before you choose the transform, the selected column is automatically assigned as the input column in the **Selected Columns** list, and you are prompted for the output column.
2. Pick the worksheet column with the data you want to standardize as the input column by clicking it in the worksheet or selecting it from the **Data for Input** drop-down list. The number or title of the selected column appear in the highlighted input row, and you are prompted for an output column.
3. Pick the column where you want the standardized variables to appear as the output column by clicking it the worksheet or selecting it from the **Data for Output** drop-down list. The number or title of the selected column appears in the highlighted output row.
4. To change your selections, select the column assignment in the Selected Columns list, then select the desired column from the worksheet or drop-down list. You can clear a column assignment by double-clicking it.
5. Click **Finish** to run the transform on the specified input column and place the results in the specified output column.

   Tip: If you specify an output column that contains data, a dialog box appears asking you if you want to erase the column contents, push the contents down, or cancel the transform. Click **Overwrite** to replace the existing column contents with the transform results. Select **Insert** to place transform results above the existing cell contents. The data from the input columns are factored together and placed in the specified output column.

Ranking Data

Use the rank transform to assign integer rank values to data. Ranking data is useful if you want know how the values are ranked, or to perform two way ANOVA on the ranks of data that fails the normality or equal variance tests. The rank transform assigns rank values to all observations in a column from smallest to largest. Equal values are tied in rank, and an averaged rank is assigned to all tied values. This rank is the average of the ranks that would have been assigned to all the tied values if they were not tied.

To rank a variable:

1. On the **Analysis** tab, in the **Transform** group, select:

   Statistical > **Rank**

   The **Rank Transform - Select Data** dialog box appears and prompts you to select an input column.

   Note: If you select a column in the worksheet before you choose the transform, the selected column is automatically assigned as the input column, and you are prompted for the output column.

2. Pick the column with the data you want to rank as the input column by clicking it in the worksheet or selecting it from the **Data for Input** drop-down list. The number or title of the selected column appears in the highlighted input row, and you are prompted for an output column.
3. Pick the column where you want the ranked variables to appear as the output column by clicking it the worksheet or selecting it from the **Data for Output** drop-down list. The number or title of the selected column appears in the highlighted output row.
4. To change your selections, select the column assignment in the Selected Columns list, then select the desired column from the worksheet or drop-down list. You can clear a column assignment by double-clicking it.
5. Click **Finish** to run the transform on the specified input column and place the results in the specified output column

   Tip: If you specify an output column that contains data, a dialog box appears asking you if you want to erase the column contents, push the contents down, or cancel the transform. Click **Overwrite** to replace the existing column contents with the transform results. Select Insert to place transform results above the existing cell contents. The data from the input columns are factored together and placed in the specified output column.

Creating Interaction Variables

Use the interaction transform to when you want to introduce an interaction variable into a multiple linear regression model, for example, a variable that takes into account the interaction between two independent variables. The interaction transform computes the product of the values in two data columns and places the results in an output column. For example, to introduce an interaction factor into the general multiple linear regression model: $y = b_0 + b_1x_1 + b_2x_2$ you could add another variable to the equation equal to x_1x_2, for example: $y = b_0 + b_1x_1 + b_2x_2 + b_3x_1x_2$

Tip: Adding an interaction variable to a multiple linear regression can induce multicollinearity. To avoid or reduce this problem, use the Center transform on the original variables, then use the centered variables to generate the interaction variable. For descriptions of independent variable interactions in multiple linear regression, you can reference any appropriate statistics reference.

To generate an interaction variable:

1. On the **Analysis** tab, in the **Transform** group, select:

   Statistical > **Interactions**

   The **Pick Columns for Interactions Transform** dialog box appears and prompts you to select an input column.

   Note: If you selected columns before you ran the transform, the selected columns are assigned as the input and output columns in the order they were selected in the worksheet.

2. Pick the first variable column with the data you want to factor into the interaction by clicking it in the worksheet or selecting it from the **Data for Input** drop-down list, then pick the second input column. The number or title of the selected column appears in the input row of the **Selected Columns** list.
3. Select the column where you want to place the interaction variable as the output column by clicking it in the worksheet or selecting it from the Data for Output drop-down list. The number or title of the selected column appears in the highlighted output row.
4. To change your selections, select the column assignment in the Selected Columns list, then select the desired column from the worksheet or drop-down list. You can clear a column assignment by double-clicking it.
5. Click **Finish** to run the transform on the specified input columns and place the results in the specified output column.

   Tip: If you specify an output column that contains data, a dialog box appears asking you if you want to erase the column contents, push the contents down, or cancel the transform. Click **Overwrite** to replace

the existing column contents with the transform results. Select **Insert** to place transform results above the existing cell contents. The data from the input columns are factored together and placed in the specified output column.

Creating Dummy (Indicator) Variables

Dummy, or indicator, variables can be used to determine if sets of data share the same constant (intercept) value, by determining if the constant is affected by conditional changes specified by the dummy variables. This can be used to determine if all the data in a simple linear regression lie on the same line, or if there are conditional dependencies on the independent variable.

Dummy variables are generally computed from indexed data columns. They are assigned to index variable data similarly to the way index values are assigned to raw data, but always correspond to specific numeric values, as determined by the index variable values and by the kind of dummy variable coding used. There are two ways to define dummy variables: reference coding and effects coding.

If the index column contains numeric values, the dummy variable transform uses the nearest whole number as the code value, and evaluates the data for the corresponding dummy variable by rounding up to the nearest whole number.

For *k* number of different index variable values (conditions of possible dependencies), the dummy variables transform creates *k*–1 dummy variables. If an index variable contains two different index values, one dummy variable column is produced, and the other is used as the reference or effect index value; if the index variable contains three different index values, two dummy variable columns are produced, and the third is used as the reference or effect value. To create a dummy variable column, the index column must contain at least two different index values. For descriptions of how to use dummy variables to detect different slopes, you can reference any appropriate statistics reference.

Reference Coding

Reference coding sets the value of all dummy variables to zero when the index variable corresponds to the indexed condition used, and codes all other values of the index variable with a 1.The referenced condition is always assigned a 0.

Tip: Use reference coding when you want the constant to be the mean of the dependent variable under a selected referenced condition, and the coefficients computed for the dummy variable(s) to reflect the changes of the constant value from reference condition dependent variable mean.

To create reference coded dummy variables:

1. If necessary, create an index column for your data. These data can consist of any numbers or strings. Each dependent variable value that falls under a different condition is indexed with a different label. For more information, see Indexing Data. Two factor and repeated measures data require additional index columns.
2. On the **Analysis** tab, in the **Transform** group, select:

   Statistical > **Dummy Variables** > **Reference Coding**

   The **Reference Transform - Select Data** dialog box appears and prompts you to select input and output columns.

   Tip: If you selected columns before you ran the transform, the selected columns are assigned as the input and output columns in the order they were selected in the worksheet.
3. Pick the column with the indexed data you want to create dummy variables for as the input column by clicking it in the worksheet or selecting it from the **Data for Input** drop-down list. The number or title of the selected column appear in the highlighted input row and you are prompted for the output column.
4. Select the destination column for the dummy variables as the output column by clicking it in the worksheet or selecting it from the Data for Output drop-down list. The number or title of the selected column appears in the highlighted output row. There should be enough empty columns to the right of the destination column to accommodate all the dummy variable columns; the number of dummy variable columns produced is one less than the number of index values (different groups).
5. To change your selections, select the column assignment in the Selected Columns list, then select the desired column in the worksheet. Select additional columns as th

7. Select the reference index value from the list to use as the reference condition; no dummy variable is created using this value (this is the condition that determines the constant value; the corresponding dummy variable values for this condition are always zero). All other index values are evaluated for the corresponding dummy variable values.
8. Click **OK**. The reference coded dummy variables are placed in as many columns as there are index values, less one. Index column values that match the condition used to evaluate the column are assigned a zero; all other values are assigned a 1. One dummy variable column is produced for each index value, except for the index value selected as the reference condition.

   Note: If you specify an output column that contains data, a dialog box appears asking you if you want to erase the column contents, push the contents down, or cancel the transform. Click **Overwrite** to replace the existing column contents with the transform results. Select **Insert** to place transform results above the existing cell contents. The data from the input columns are factored together and placed in the specified output column.

Effects Coding

In effects coding, the dummy variables are coded with -1, 0, and 1. The reference condition is always coded with a -1. The value of other dummy variables is set to zero when the index variable corresponds to the indexed condition used, and set to 1 for all other values of the index variable.

Important: Use effects coding when you want the constant term to be computed using the value of the dependent variable under all indexed conditions, and you want the coefficients of the dummy variables to quantify the size of changes from this overall mean.

To create effects coded dummy variables:

1. If necessary, create an index column for your data. This data can consist of any numbers or strings. Each dependent variable value that falls under a different condition is indexed with a different label. For more information, see Indexing Data. Two factor and repeated measures data require additional index columns.
2. On the **Analysis** tab, in the **Transform** group, select:

   Statistical > **Dummy Variables** > **Effects Coding**

   The **Effects Transform - Select Data** dialog box appears and prompts you select an input column.
3. If you selected columns before you ran the transform, the selected columns are assigned as the input and output columns in the order they were selected in the worksheet.
4. Pick the column with the indexed data you want to create dummy variables for as the input column by clicking it in the worksheet or by selecting it from the **Data for Input** drop-down list. The number or title of the selected column appears in the highlighted input row, and you are prompted for the output column.
5. To change your selections, select the column assignment in the **Selected Columns** list, then select the desired column from the worksheet or drop-down list. You can clear a column assignment by double-clicking it.
6. Pick the destination column for the dummy variable column(s) as the output column by clicking it in the worksheet or selecting it from the **Data for Output** drop-down list. The number or title of the selected column appears in the highlighted output row of the **Selected Columns** list. There should be enough empty columns to the right of the destination column to accommodate all the dummy variable columns; the number of dummy variable columns produced is one less than the number of index values (different groups).
7. Click **Finish** to run the transform and open the **Select Reference Index** dialog box.
8. Select the reference index value from the list to use as the reference; no dummy variable is created for this value, and the corresponding dummy variable values for this condition are always -1. All other dummy variable values are set to 1 for the corresponding index variable values.
9. Click **OK**. Values in the index column that match the index value used to evaluate the column are assigned a zero. Index values that match the reference condition are assigned -1. All other values are set to 1. One dummy variable column is produced for each index value, except the index selected as the reference condition.

   Tip: If you specify an output column that contains data, a dialog box appears asking you if you want to erase the column contents, push the contents down, or cancel the transform. Click **Overwrite** to replace the existing column contents with the transform results. Select **Insert** to place transform results above the

existing cell contents. The data from the input columns are factored together and placed in the specified output column.

If you are creating dummy variables for a two factor or repeated measures problem, create dummy variables for all remaining index columns.

Performing a Regression Using Dummy Variables

The equation used to evaluate the effect of a condition on the regression model constant is:

where y is the dependent variable, x is the independent variable, k is the number of conditions that may affect the constant, d_1, d_2, d_{k-1} are the dummy variables, and b_0, b_1, b_2, b_{k-1} are the coefficients.

To perform a Multiple Linear Regression using dummy variables:

1. On the **Analysis** tab, in the **Statistics** group, select:

   Tests > **Regression** > **Multiple Linear Regression**

   The **Multiple Linear Regression - Select Data** dialog box appears.
2. Select the dependent variable column, then select the original independent variable and all the dummy variables as the independent variables.
3. Click **Finish** to run the regression on the selected columns.
4. Compare the results to the original Simple Linear Regression. If the prediction is significantly better, you should consider performing a simple linear regression on the different conditions separately.

   You can use dummy variables to convert analysis of variance problems into regression problems. For more information on how to do this, you can reference any appropriate statistics reference.

> **Tip:** If you specify an output column that contains data, a dialog box appears asking you if you want to erase the column contents, push the contents down, or cancel the transform. Click **Overwrite** to replace the existing column contents with the transform results. Select **Insert** to place transform results above the existing cell contents. The data from the input columns are factored together and placed in the specified output column.

Creating Lagged Variables

The lagged variables transformation lags the observations in one column by one row, by inserting a missing value to the first row of the data and removing the last value; the overall column size remains constant.

Lagged variables are commonly used to create time series models, when the effect of an independent variable on the dependent variable corresponds more appropriately to the value of the dependent variable at a later time.

To create lagged variables:

1. On the **Analysis** tab, in the **Transform** group, select:

   Statistical > **Lagged Variables**

   The **Lagged Variable Transform - Select Data** dialog box appears and prompts you to pick an input column.
2. Pick the column with the data you want to lag as the input column by clicking it in the worksheet or selecting it from the **Data for Input** drop-down list. The number or title of the selected column appears in the highlighted input row of the **Selected Columns** list, and you are prompted for an output column.
3. Pick the column where you want the lagged variables to appear as the output column by clicking it in the worksheet or selecting it from the **Data for Output** drop-down list. The number or title of the selected column appears in the highlighted output row of the **Selected Columns** list.

> **Tip: To change your selections,** select the column assignment in the **Selected Columns** list, then select the desired column from the worksheet or drop-down list. You can clear a column assignment by double-clicking it.

4. Click **Finish** to run the transform on the specified input column and place the results in the specified output column.

   ⚠ **Attention:** If you specify an output column that contains data, a dialog box appears asking you if you want to erase the column contents, push the contents down, or cancel the transform.

5. Select **Overwrite** to replace the existing column contents with the transform results. Select **Insert** to place transform results above the existing cell contents. The data from the source column is lagged by one row and placed in the specified column.
6. Repeat this transform if you need to lag the data by additional rows.

Filtering Strings and Numbers

You can isolate specified groups of data using both numeric and text filters. The filter transform operates by selecting only the rows that correspond to specified numbers or labels in a key column, then placing these rows and the corresponding data in new columns.

You can sort data according to a numeric range of a key column or according to the text label in a key column. These columns are usually factor or subject index columns for indexed data.

To sort numerically or using text:

1. On the **Analysis** tab, in the **Transform** group, select:

   Statistical > **Filter**

   The **Filter Data Transform - Select Data** dialog box appears and prompts you to pick a key column.

   ℹ **Tip:** If you select columns before you choose the missing values transform, the selected columns are assigned as the input and output column is the order they were selected in the worksheet.

2. Pick the key column to filter by clicking it in the worksheet or selecting it from the **Data for Key** drop-down list. This is the column you want to apply the sorting filter to. The number or title of the selected column appears in the highlighted key row, and you are prompted for an output column
3. Pick the column where you want the results of the key column to appear as the output column by clicking it in the worksheet or selecting it from the **Data for Output** drop-down list. The number or title of the selected column appear in the highlighted output row, and you are prompted for an input column.
4. Pick in the columns that contain the corresponding data to be filtered along with the key column as the input columns, then pick their corresponding output columns by clicking them in the worksheet or selecting them from the drop-down lists. You can pick as many input columns as desired, and you must pick an output column for every input column.

   ℹ **Tip: To change your selections,** select the column assignment in the **Selected Columns** list, then select the desired column from the worksheet or drop-down list. You can clear a column assignment by double-clicking it.

5. Click **Finish** to run the Filter transform.
   The **Set Filter** dialog box appears.
6. Select **Numeric Filter** to sort the key column data according to a numeric range. Specify the upper and lower bounds of the values to filter in the **Upper Bound** and **Lower Bound** boxes.
7. Select **Text Filter** to sort the key column data according to a text label in the key column. Enter the string exactly as it appears in the worksheet in the **Key Label** box and click **OK** when you have specified the appropriate filter.

   ⚠ **Attention:** If you specify an output column that contains data, a dialog box appears asking you if you want to erase the column contents, push the contents down, or cancel the transform.

Generating Random Numbers

You can generate either uniform or normally distributed random numbers. These commands perform identically to the random and Gaussian user-defined transform functions.

Uniformly Distributed Random Numbers

To generate uniformly distributed random numbers:

1. On the **Analysis** tab, in the **Transform** group, select:

   Statistical > **Random Numbers** > **Uniform**

   The **Uniform Random Transform - Select Data** dialog box appears and prompts you for an output column.

   Important: Input columns are not selected for the random number transform.

2. Pick the column where you want the random numbers to appear as the output column by clicking it in the worksheet or selecting it from the **Data for Output**

7. Click **OK** when finished. The random numbers are generated according to your specifications and appear in the selected output.

   ⚠ **Attention:** If you specify an output column that contains data, a dialog box appears asking you if you want to erase the column contents, push the contents down, or cancel the transform. Click **Overwrite** to replace the existing column contents with the transform results. Select **Insert** to place transform results above the existing cell contents. The data from the input columns are factored together and placed in the specified output column.

Translating Missing Value Codes

The missing data transformation converts specified values in selected columns to the SigmaPlot missing value double-dash indicator ("--"). Use this transform to translate bad or missing value codes from other data formats to the SigmaPlot format. You can also use this transform to convert all incidences of a bad observation to missing values.

To convert all occurrences of a string to "--"

1. On the **Analysis** tab, in the **Transform** group, select:

   Statistical > **Missing Values**

   The **Missing Values Transform - Select Data** dialog box appears and prompts you to select an input column.

   ℹ **Tip:** If you select columns before you choose the missing values transform, the selected columns are assigned as the input and output columns in the order they were selected in the worksheet.

2. Pick the columns with the strings you want convert to missing values as the input column by clicking it in the worksheet or selecting it from the Data for Input drop-down list; then the corresponding output column.

   ℹ **Tip:** You must pick an output column for every input column you select. You can pick as many input columns as desired. The number or title of the selected columns appear in the highlighted input and output rows in the Selected Columns list.

3. Select **Overwrite** to replace the existing column contents with the transform results. Select **Insert** to place transform results above the existing cell contents.
4. **To change your selections,** select the column assignment in the **Selected Columns** list, then select the desired column from the worksheet or drop-down list. You can clear a column assignment by double-clicking it.
5. Click **Finish** to run the transform and open the **Missing Value Transform** dialog box.
6. Specify the string to replace with missing value symbols. Enter the string exactly as it appears in the worksheet, or select the string from the drop-down list.
7. Click **OK** when finished. The specified symbols are converted to missing values.

   ⚠ **Attention:** If you specify an output column that contains data, a dialog box appears asking you if you want to erase the column contents, push the contents down, or cancel the transform.

Quick Transforms

Use the **Quick Transform** dialog box to execute simple, one-line mathematical functions to modify one or more columns of data. No knowledge of complex programming is required.

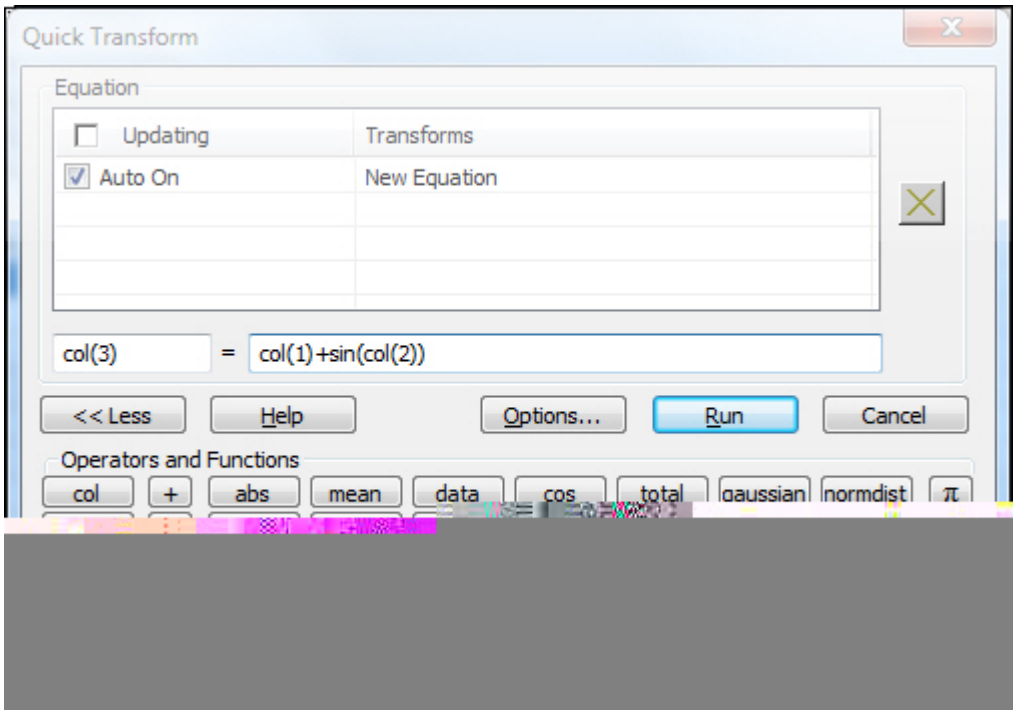
You can run Quick Transforms on an individual cell, an entire column, or a block of data.

Performing Quick Transforms

To run a Quick Transform:

1. On the **Analysis** tab, click **Quick Transform**.

⚠ **Attention:** Hover your mouse in the image below over operators and functions for brief descriptions, or click for full explanations.



1. The abs function returns the absolute value for each number in the specified range.
2. The exp function returns a range of values consisting of the number e raised to each number in the specified range. This is numerically identical to the expression e^(numbers).
3. The ln function returns a value or range of values consisting of the natural logarithm (base e) of each number in the specified range.
4. The log function returns a value or range of values consisting of the base 10 logarithm of each number in the specified range.
5. The sqrt function returns a value or range of values consisting of the square root of each value in the specified range. Numerically, this is the same as {numbers}^0.5, but uses a faster algorithm.
6. The mod function returns the modulus (the remainder from division) for corresponding numbers in numerator and divisor arguments. This is the real (not integral) modulus, so both ranges may be nonintegral values.
7. The mean function returns the average of the range specified. Use this function to calculate column averages (as opposed to using the avg function to calculate row averages).
8. The median function returns the median of the range specified. The median is a number that is both less than or equal to half and greater than or equal to half of the values in the data set.
9. The min function returns the smallest number in the range specified.
10. max on page 204
11. The stddev function returns the standard deviation of the specified range.
12. The stderr function returns the standard error of the mean of the specified range.
13. The data function generates a range of numbers from a starting number to an end number, in specified increments.
14. The int function returns a number or range of numbers equal to the largest integer less than or equal to each corresponding number in the specified range. All numbers are rounded down to the nearest integer.
15. The prec function rounds a number or range of numbers to the specified number of significant digits, or places of significance. Values are rounded to the nearest integer; values of exactly 0.5 are rounded up.
16. The round function rounds a number or range of numbers to the specified decimal places of accuracy. Values are rounded up or down to the nearest integer; values of exactly 0.5 are rounded up by default.

17. The nth function returns a sampling of a provided range, with the frequency indicated by a scalar number. The result always begins with the first entry in the specified range.

18. The avg function averages the numbers across corresponding ranges, instead of within ranges. The resulting range is the row-wise average of the range arguments. Unlike the mean function, avg returns a range, not a scalar.

19. This function returns ranges consisting of the cosine of each value in the argument given.

20. This function returns ranges consisting of the sine of each value in the argument given.

21. This function returns ranges consisting of the tangent of each value in the argument given. This and other trigonometric functions can take values in radians, degrees, or grads.

22. This function returns the inverse of the corresponding trigonometric function.

23. This function returns the inverse of the corresponding trigonometric function.

24. This function returns the inverse of the corresponding trigonometric function.

25. The function total returns a single value equal to the total sum of all numbers in a specified range. Numerically, this is the same as the last number returned by the sum function.

26. The function sum returns a range of numbers representing the accumulated sums along the list. The value of the number is added to the value of the preceding cumulative sum.

27. The diff function returns a range or ranges of numbers which are the differences between a given number in a range and the preceding number. The value of the preceding number is subtracted from the value of the following number.

28. The factorial function returns the factorial of a specified range.

29. The choose function determines the number of ways of choosing r objects from n distinct objects without regard to order.

30. The transform function rgbcolor takes arguments r, g, and b between 0 and 255 and returns the corresponding color to cells in the worksheet. This function can be used to apply custom colors to any element of a graph or plot that can use colors chosen from a worksheet column.

31. This function generates a specified number of normally (Gaussian or “bell” shaped) distributed numbers from a seed number, using a supplied mean and standard deviation.

32. This function generates a specified number of uniformly distributed numbers within the range. Rand and rnd are synonyms for the random function.

33. The runavg function produces a range of running averages, using a window of a specified size as the size of the range to be averaged. The resulting range is the same length as the argument range.

34. The inv function generates the inverse matrix of an invertible square matrix provided as a block.

35. The fft function finds the frequency domain representation of your data using the Fast Fourier Transform.

36. The inverse fft function (invfft) takes the inverse Fast Fourier Transform (fft) of the data produced by the fft to restore the data to its new filtered form.

37. This function is the cumulative normal (or Gaussian) distribution function. It returns the probability that a normal random variable is less than a specified independent variable value.

38. This function is the inverse cumulative normal (or Gaussian) distribution function. The probability that a normally distributed random variable is less than the return value is equal to the argument you specify.

39. This function is Student’s T-distribution function. It returns the probability that a T-distributed random variable is less than a specified independent variable value.

40. This function is the inverse of Student’s T-distribution function. The probability that a T-distributed random variable is less than the return value is equal to the argument you specify.

41. This function is the F-distribution function. It returns the probability that an F distributed random variable is less than a specified independent variable value.

42. This function is the inverse F-distribution function. The probability that an F-distributed random variable is less than the return value is equal to the argument you specify.

43. The col function returns all or a portion of a worksheet column, and can specify a column destination for transform results.

44. The cell function returns the contents of a cell in the worksheet, and can specify a cell destination for transform results.

45. The block function returns a block of cells from the worksheet, using a range specified by the upper left and lower right cell row and column coordinates.

Figure 3: The transform in this example applies the sine function to each entry in Column 2 and then adds the results row-wise to the entries of Column 1. The final results will appear in Column 3.

The **Quick Transform** dialog box appears with an array of functions. These provide immediate access to frequently used transforms.

2. In the edit box on the left, specify the column or block of cells of the worksheet in which you want the results of the transform to appear.
3. In the edit box on the right, enter a one-line transform using any functions defined in the transform language.

> **Tip:** Click **< < Less** to close the **Functions**.

4. You cannot edit these equations. To delete an equation, select it from the list and then click the Delete button to the right of the list.

> **Restriction:** You cannot run transforms on date and time columns. To use date and time data, you must first convert the data to numeric data, run the transform, and then convert the column back to date and time data. For more information, see Switching Between Date and Time and Numeric Display.

Setting Trigonometric Units for Quick Transforms

You can quickly set trigonometric units in the Quick Transforms Options dialog box.

1. Make sure that you have a worksheet in view.
2. On the **Analysis** tab, click **Quick Transform**.
3. On the **Quick Transform** dialog box, click **Options**.

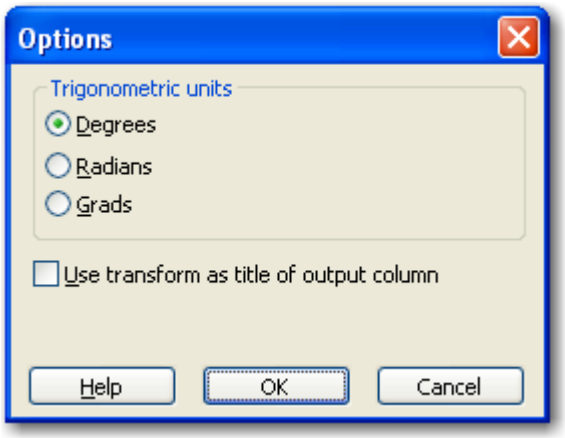


Figure 4: Click the Options button to set Trigonometric units.

4. Select the appropriate trigonometric units for calculating trigonometric functions.

Setting a Quick Transform as a Column Title

When you run either a User-Defined or Quick Transform, its results will create or overwrite data. When running a Quick Transform, you can apply the name of the transform as a column title to reflect its type of results.

1. Make sure that you have a worksheet in view. .
2. On the **Analysis** tab, click **Quick Transform**.
3. On the **Quick Transform** dialog box, click **Options**.
4. Select **Use transform as the title of the output column**.

5. Click **OK** to accept these options and close the dialog box. Here are some examples of column title possibilities:
   - Using a Quick Transform of `col(3) = col(1)+col(2)`, results in the column title for column 3 of: **col(1)+col(2)**.
   - Using a Quick Transform of `col(4) = col(2)+col(3)`, results in the column title for column 4 of: **col(2)+col(3)**.
6. Click **Run** to run the transform. Results appear in the cell, block of data, or column specified in the left drop-down list.

User-Defined Transforms

Modify and manipulate worksheet data by entering SigmaPlot's extensive mathematical transformation language into the **User-Defined Transform** dialog box. Use transforms to create new data by performing functions on existing data, or generate calculated or random data, which can then be placed in worksheet columns. For more information, see Transform Operators on page 25.

The first step to transform worksheet data is to enter the desired equations in the edit box of the **User-Defined Transform** dialog box. For more information, see Creating User-Defined Functions. If no previously entered transform equations exist, the edit box is empty; otherwise, the last transform entered appears.

Select the edit box to begin entering transform instructions. As you enter text into the transform edit box, the box scrolls down to accommodate additional lines. For more information, see Entering Transforms on page 20.

You can enter up to 100 lines of equations, either on separate lines or on the same line.

Creating a User-Defined Transform

1. View the worksheet.
2. On the **Analysis** tab, in the **Transform** group, click **User-Defined**.

   The **User-Defined Transform** dialog box appears.

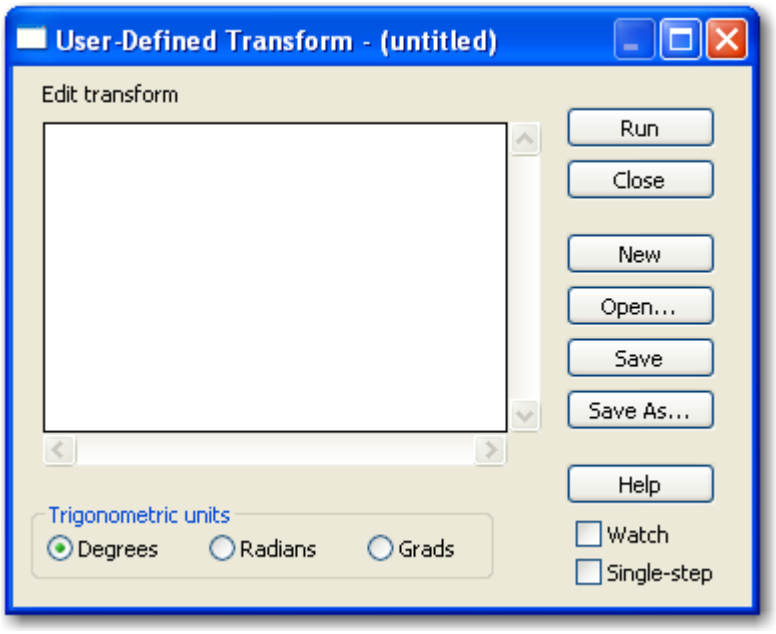


Figure 5: User-Defined Transform Dialog Box

3. Type transform instructions into the **Edit Transform** field. You can enter up to 32,000 characters.

4. Click **Run**.

> **Tip:** You can save the contents of the transform window to a file. Since this is a text file, you can view or print these files using any word processor. You can open previously saved transforms in the transform window for execution or modification.

All transform files have the extension of **.xfm** in the Transforms folder. To view these files, click the **Open** button in the **User-Defined Transforms** dialog box and open a transform file. A library of transform results is named **Xfms.jnb** in the Transform folder. These transform examples also include a sample SigmaPlot graph file displaying the results of the transform.

Transform Syntax and Structure

Use standard syntax and equations when defining user-defined transforms. This section discusses the basics and the details for entering transform equations.

Transform Syntax

Enter transforms as equations with the results placed to the left of the equal sign (=) and the calculation placed to the right of the equal sign. Results can be defined as either variables (which can be used in other equations), or as the worksheet column or cells where results are to be placed.

Entering Transforms

To type an equation in the transform edit box, click in the edit box and begin typing. When you complete a line, press Enter to move the cursor to the first position on the next line.

You can leave spaces between equation elements: `x = a+b` is the same as `x = a + b`. However, you may find it necessary to conserve space by omitting spaces. Blank lines are ignored so that you can use them to separate or group equations for easier reading.

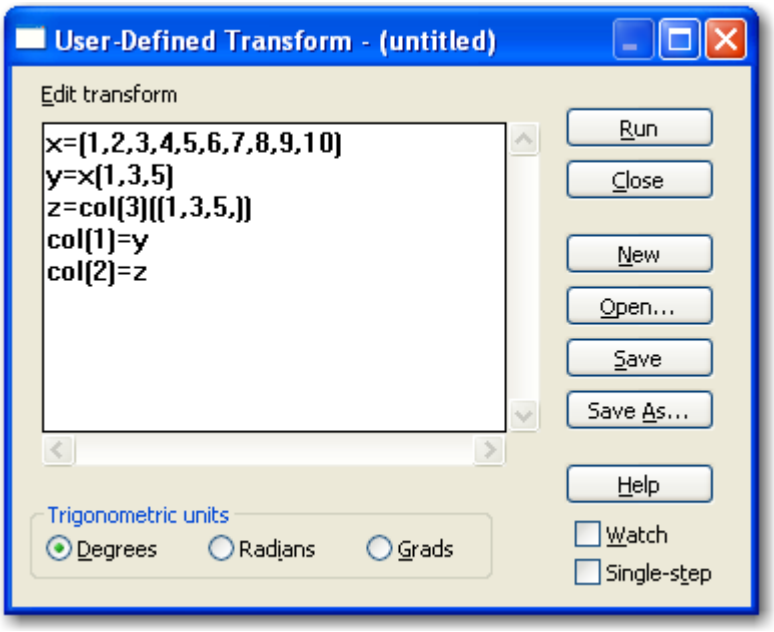


Figure 6: Typing Equations into the Edit Window

If the equation requires more than one line, you may want to begin the second and any subsequent lines indented a couple of spaces (press the space bar before typing the line). Although this is not necessary, indenting helps distinguish a continuing equation from a new one.

> **Tip:** You can resize the transform dialog box to enlarge the edit box. You can press Ctrl+X, Ctrl+C, and Ctrl +V to cut, copy, and paste text in the edit window.

Transforms are limited to a maximum of 100 lines. Note that you can enter more than one transform statement on a line; however, this is only recommended if space is a premium.

Important: Use only parentheses to enclose expressions. Curly brackets and square brackets are reserved for other uses.

Commenting on Equations

To enter a comment, type an apostrophe (') or a semicolon (;), then type the comment to the right of the apostrophe or semicolon. If the comment requires more than one line, repeat the apostrophe or semicolon on each line before continuing the comment.

Sequence of Expression

SigmaPlot generally solves equations regardless of their sequence in the transform edit box; however, the col function (which returns the values in a worksheet column) depends on the sequence of the equations, as shown in the following example.

Example

The sequence of the equations:

```
col(1)=col(4)^alpha
col(2)=col(1)*theta
```

must occur as shown. The second equation depends on the data produced by the first. Reversing the order produces different results. To avoid this sequence problem, assign variables to the results of the computation, then equate the variables to columns:

```
x=col(4)
y=x^alpha
z=y*theta
col(1)=y
col(2)=z
```

The sequence of the equations is now unimportant.

Transform Components

Transform equations consist of *variables* and *functions*. *Operators* are used to define variables or apply functions to scalars and ranges. A *scalar* is a single worksheet cell, number, missing value, or text string. A *range* is a worksheet column or group of scalars.

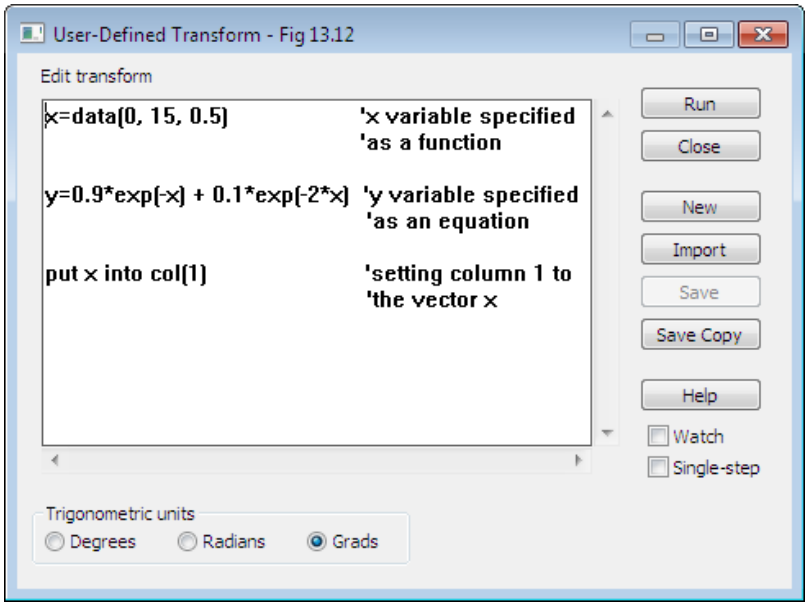


Figure 7: Examples of the Transform Equation Elements Typed into the Transform Window

Variables

You can define *variables* for use in other equations within a transform. Variable definition uses the following form:

```
variable = expression
```

Variable names must begin with a letter. After that, they can include any letter or number, or the underscore character (_). Variable names are case sensitive – an "A" is not the equivalent of an "a." Once a variable has been defined by means of an expression, that variable cannot be redefined within the same transform.

Functions

A *function* is similar to a variable, except that it refers to a general expression, not a specific one, and thus requires arguments. The syntax for a function declaration is

```
function(argument 1,argument 2,...) = expression
```

where *function* is the name of the function, and one or more argument names are enclosed in parentheses. Function and argument names must follow the same rules as variable names.

User-Defined Functions. Frequently used functions can be copied to the Clipboard and pasted into the transform window.

Constructs

Transform *constructs* are special structures that allow more complex procedures than functions. Constructs begin with an opening condition statement, followed by one or more transform equations, and end with a closing statement. The available constructs are for loops and if...then...else statements.

Operators

A complete set of arithmetic, relational, and logic operators are provided. Arithmetic *operators* perform simple math between numbers. Relational operators define limits and conditions between numbers, variables, and equations. Logic operators set simple conditions for if statements. For more information, see Transform Operators on page 25.

Numbers

You can enter numbers as integers, in floating point style, or in scientific notation. All numbers are stored with 15 figures of significance. Use a minus sign in front of the number to signify a negative value.

Missing values, represented in the worksheet as a pair of dashes, are considered non-numeric. All arithmetic operations which include a missing value result in another missing value.

To generate a missing value, divide zero by zero.

Example

If you define:

```
missing = 0/0
```

the operation:

```
size({1,2,3,missing})
```

returns a value of 4.0. (The size function returns the number of elements in a range, including labels and missing values.)

The transform language does not recognize two successive dashes; for example, the string {1,2,3,--} is not recognized as a valid range. Dashes are used to represent missing values in the worksheet only.

Strings, such as text labels placed in worksheet cells, are also non-numeric information. To define a text string in a transform, enclose it with double quotation marks.

As with missing values, strings may not be operated upon, but are propagated through an operation. The exception is for relational operators, which make a lexical comparison of the strings, and return true or false results accordingly.

Scalars and Ranges

The transform language recognizes two kinds of elements: *scalars* and *ranges*. A scalar is any single number, string, or missing value. Anything that can be placed in a single worksheet cell is a scalar.

A range (sometimes called a vector or list) is a one-dimensional array of one or more scalars. Columns in the worksheet are considered ranges.

Ranges can also be defined using curly bracket ({}) notation. The range elements are listed in sequence inside the brackets, separated by commas. Most functions which accept scalars also accept ranges, unless specifically restricted. Typically, whatever a function does with a scalar, it does repeatedly for each entry in a range. A single function can operate on either a cell or an entire column.

Example 1

The entry:

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

produces a range of five values, from 1 through 5.

Example 2

The operation:

```
{col(1), col(2)}
```

concatenates columns 1 and 2 into a single range. Note that elements constituting a range need not be of the same type, i.e., numbers, labels and missing values.

Example 3

The entry:

```
{x,col(4)*3,1,sin(col(3))}
```

also produces a range.

Array References

Individual scalars can be accessed within a range by means of the square bracket ([ ]) constructor notation. If the bracket notation encloses a range, each entry in the enclosed range is used to access a scalar, resulting in a new range with the elements rearranged.

Example

For the range:

```
x = {1.4,3.7,3.3,4.8}
```

the notation:

```
x[3]
```

returns 3.3, the third element in the range. The notation:

```
x[{4,1,2}]
```

produces the range {4.8,1.4,3.7}. The constructor notation is not restricted to variables: any expression that produces a range can use this notation.

Example

The operation:

```
col(3)[2]
```

produces the same result as col(3,2,2), or cell(3,2). The notation:

```
{2,4,6,8}[3]
```

produces 6. If the value enclosed in the square brackets is also a range, a range consisting of the specified values is produced.

Example

The operation:

```
col(1)[{1,3,5}]
```

produces the first, third, and fifth elements of column 1.

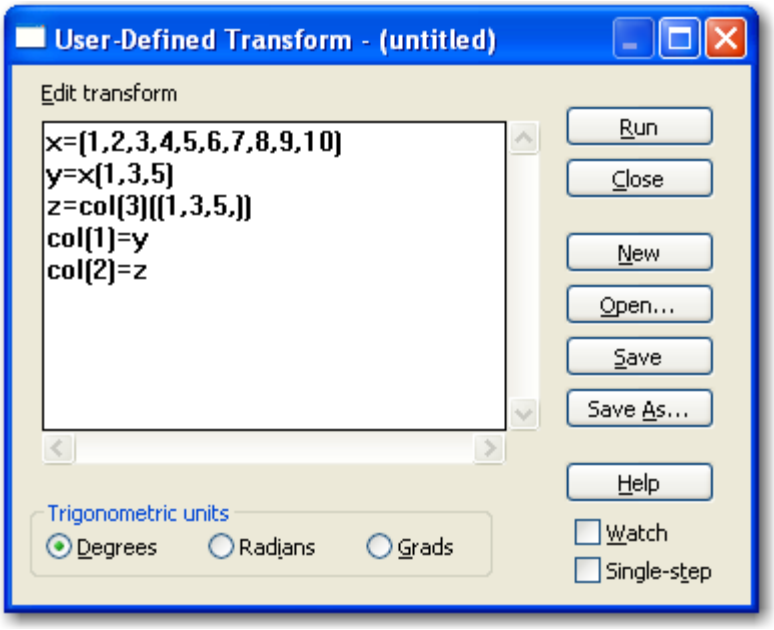


Figure 8: Range and Array Reference Operations Typed into the User Defined Transform Window

Transform Operators

Transforms use operators to define variables and apply functions. A complete set of arithmetic, relational, and logical operators are provided.

Order of Operation

The order of precedence is consistent with P.E.M.A. (Parentheses, Exponentiation, Multiplication, and Addition) and proceeds as follows, except that parentheses override any other rule:

- Exponentiation, associating from right to left.
- Unary minus.
- Multiplication and division, associating from left to right.
- Addition and subtraction, associating from left to right.
- Relational operators.
- Logical negation.
- Logical and, associating from left to right.
- Logical or, associating from left to right.

This list permits complicated expressions to be written without requiring too many parentheses.

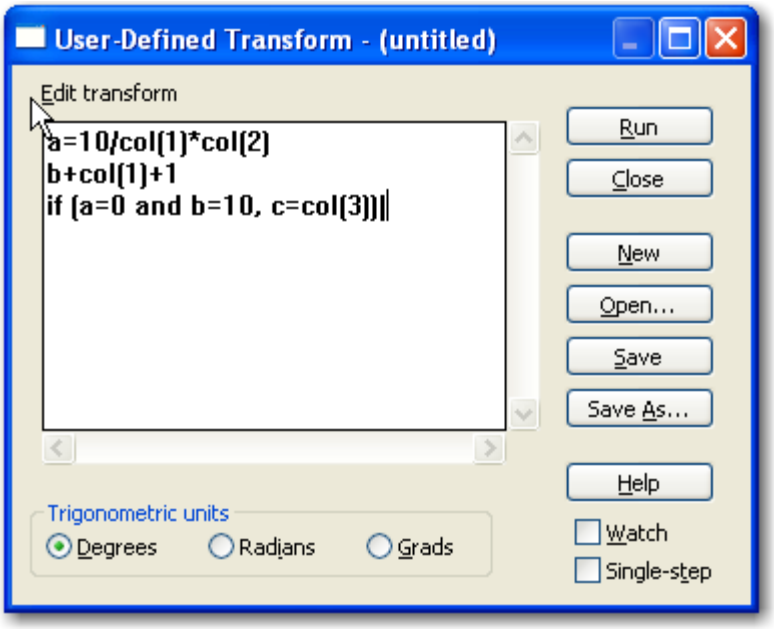


Figure 9: Examples of Transform Operators

Example

The statement:

```
a<10 and b<5
```

groups to (a<10) and (b<5), not to (a<(10 and b))<5.

Important: Only parentheses can group terms for processing. Curly and square brackets are reserved for other uses.

Operations on Ranges

The standard arithmetic operators—addition, subtraction, multiplication, division, and exponentiation—follow basic rules when used with scalars. For operations involving two ranges corresponding entries are added, subtracted, etc., resulting in a range representing the sums, differences, etc., of the two ranges.

If one range is shorter than the other, the operation continues to the length of the longer range, and missing value symbols are used where the shorter range ends.

For operations involving a range and a scalar, the scalar is used against each entry in the range.

Example

The operation:

```
col(4)*2
```

produces a range of values, with each entry twice the value of the corresponding value in column 4.

Arithmetic Operators

Arithmetic operators perform arithmetic between a scalar or range and return the result.

+	Add
-	Subtract (also signifies unary minus)
*	Multiply

/	Divide
^ or **	Exponentiate

Multiplication must be explicitly noted with the asterisk. Adjacent parenthetical terms such as `(a+b) (c-4)` are not automatically multiplied.

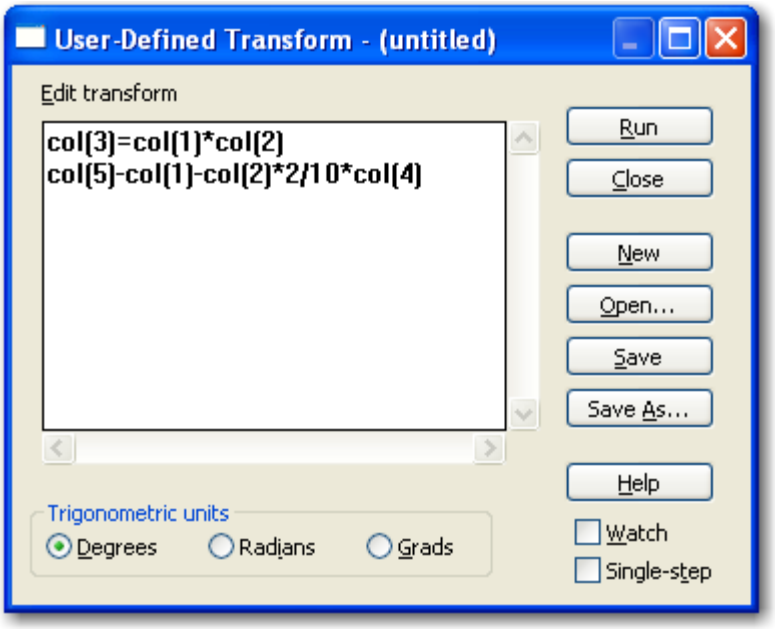


Figure 10: Examples of arithmetic operators

Relational Operators

Relational operators specify the relation between variables and scalars, ranges or equations, or between user-defined functions and equations, establishing definitions, limits and/or conditions.

= or .EQ.	Equal to
> or .GT.	Greater than
>= or .GE.	Greater than or equal to
< or .LT.	Less than
<= or .LE.	Less than or equal to
<>, !=, #, or .NE.	Not equal to

The alphabetic characters can be entered in upper or lower case.

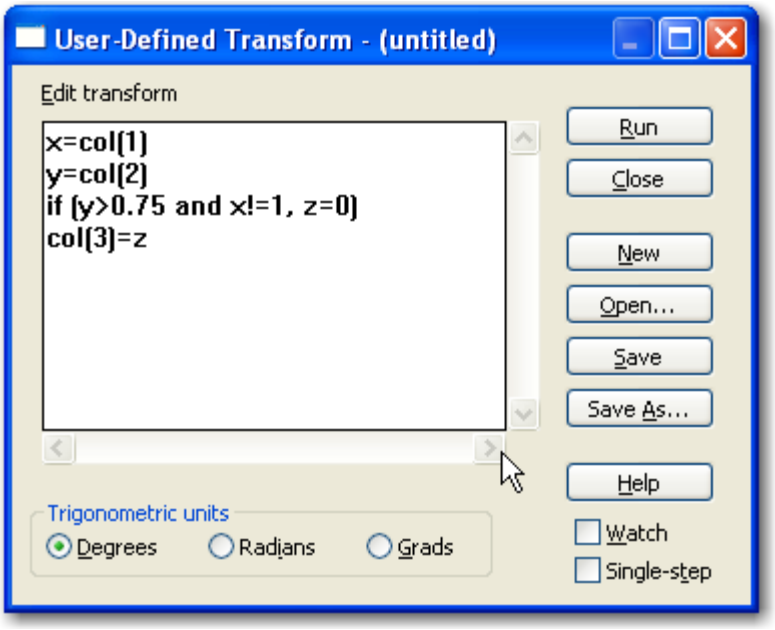


Figure 11: Examples of rational and logical operators.

Logical Operators

Logical operators are used to set the conditions for if function statements.

and, &	Intersection
or, |	Union
not, ~	Negation

Transform Examples

Many mathematical transform examples, along with appropriate graphs and worksheets are included with SigmaPlot. This chapter describes the data transform examples and the graphing transform examples provided. Each description contains the text of the transform and, where applicable, a graph displaying the possible results of the transform. All of these examples are available in the **Transforms** directory that was installed when you installed SigmaPlot.

To find the Transforms directory:

1. Click the **Main Button**, and then click **Open**.
2.

- Count
- If
- Total
- Mean
- {...} (constructor notation)

To use the One Way ANOVA transform:

1. Make sure your original Y data is in column 2. Perform the desired regression using the Regression Wizard, and save your Predicted values (fitted Y data) in column 3, and Parameters (the regression coefficients) in column 4. For more information, see Nonlinear Regression on page 79.
2. Press **F10** to open the **User-Defined Transform** dialog box, then click **Open** to open the ANOVA.XFM transform file in the XFMS directory. The ANOVA transform appears in the edit window.
3. Click **Run**. The ANOVA results are placed in columns 5 through 9, or beginning at the column specified with the ANOVA variable.

Area Beneath a Curve Using Trapezoidal Rule

This transform computes the area beneath a curve from X and Y data columns using the trapezoidal rule for unequally spaced X values. The algorithm applies equally well to equally spaced X values.

This transform uses an example of the diff function.

To use the Area Under Curve transform:

1. Place your X data in column 1 and your Y data in column 2. If your data has been placed in other columns, you can specify these columns after you open the AREA.XFM file. You can use an existing or new worksheet.
2. Click **Run**. The area is placed in column 3 or in the column specified with the res variable.

Bivariate Statistics

This transform takes two data columns of equal length and computes their means, standard deviations, covariance, and correlation coefficient. The columns must be of equal length.

The Bivariate transform uses examples of these transform functions

- mean
- stddev
- total

To use the Bivariate transform:

1. Place your X data in column 1 and your Y data in column 2. If your data has been placed in other columns, you can specify these columns after you open the BIVARIAT.XFM transform file. You can enter data into an existing worksheet or a new worksheet.
2. Click **Run**. The results are placed in columns 3 and 4, or beginning in the column specified with the res variable.

Differential Equation Solving

This transform can be used to solve user-defined differential equations. You can define up to four first order equations, named fp1(x_1,y_1,y_2,y_3,y_4) through fp4(x_1,y_1,y_2,y_3,y_4). Set any unused equations = 0.

To solve a firs order differential equation:

1. Open a new worksheet; this transform requires a clean worksheet to work correctly. For more information, see Opening Worksheets .
2. Open the **User-Defined Transforms** dialog box. For more information, see Transform Function Descriptions on page 161.
3. Open the DIFFEQN.XFM transform file in the XFMS directory. The Differential Equation Solving transform appears in the edit window.
4. Scroll to the **Number of Equations** section and enter a value for the neqn variable. This is the number of equations you want to solve, up to four.

5. Scroll down to the Differential Equations section, and set the `fp1` through `fp4` functions to the desired functions. Set any unused equations = `0`. If only one first order differential equation is used, then only the `fp1` transform equation is used and `fp2`, `fp3`, and `fp4` are set to `0`. For example, if you only wanted to solve the differential equation:

   you would enter:

```
fp1(x,y1,y2,y3,y4) = -a*y1
fp2(x,y1,y2,y3,y4) = 0
fp3(x,y1,y2,y3,y4) = 0
fp4(x,y1,y2,y3,y4) = 0
```

6. Scroll down to the Initial Values heading and set the nstep variable to the number of integration (X variable) steps you want to use. The more steps you set, the longer the transform takes.
7. Set the initial X value x0, final X value x1, and the Y1 through Y4 values (placed in cells (2,1) through (5,1)). If you are not using a y1 value, set that value to zero (0). For example, for the single equation example above, you could enter:

```
x0 = 0              ;initial x
x1 = 1              ;final x
cell(2,1) = 1       ;y1 initial value
cell(3,1) = 0       ;y2 initial value
cell(4,1) = 0       ;y3 initial value
cell(5,1) = 0       ;y4 initial value
```

8. Click **Run**. The results output is placed in columns 1 through neqn+1.
9. To graph your results, create a Line Plot graphing column 1 as your X data and columns 2 through 5 as your Y data. Creating 2D Plots with Multiple Curves

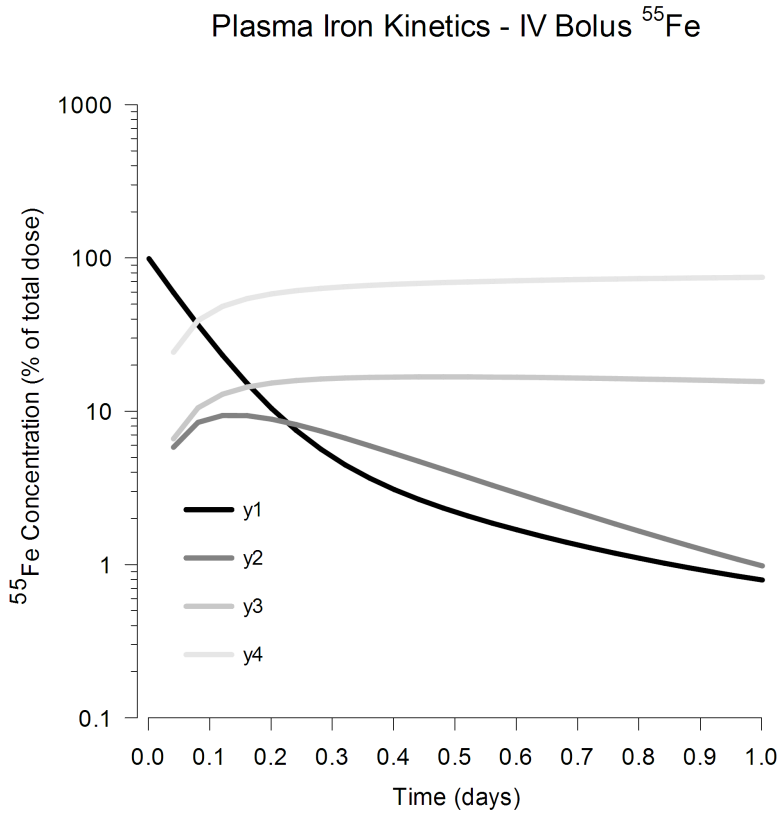


Figure 12: Differential Equation Graph

F-test to Determine Statistical Improvement in Regressions

This transform compares two equations from the same family to determine if the higher order provides a statistical improvement in fit.

Often it is unclear whether a higher order model fits the data better than a lower order. Equations where higher orders may produce better fits include: simple polynomials of different order, the sums of exponentials for transient response data, and the sums of hyperbolic functions for saturation ligand binding data.

F_TEST.XFM uses the residuals from two regressions to compute the sums of squares of the residuals, then creates the F statistic and computes an approximate P value for the significance level.

You can try this transform out on the provided sample graph, or run it on the residuals produced by your own regression sessions. Residuals are saved to the worksheet by the Regression Wizard.

1. **To use the provided sample data and graph,** open the F-test worksheet and graph in the XFMS.JNB notebook. The worksheet contains raw data in columns 1 and 2, and curve fit results for the two competitive binding models in columns 3-5 and 6-8. The graph plots the raw data and the two curve fits.
2. **To use your own data,** enter the XY data to be curve fit in columns 1 and 2, respectively. Select the first curve fit equation and use it to fit the data, place the parameters, fit results and residuals in the first empty columns (3-5). Run the second curve fit and place the results in columns 6-8 (the default). If desired, create graphs of these results using the wizard.
3. Press **F10** to open the **User-Defined Transform** dialog box, then open the F_TEST.XFM transform file. Specify n1 and n2, the number of parameters in the lower and higher order functions. In the example provided, these are 3 and 5, respectively.
   If necessary, specify cs1 and cs2, the column locations for the residuals of each curve fit, and cres, the first column for the two column output.
4. Click **Run**. The F-test value and corresponding P value are placed into the worksheet. If $P < 0.05$, you can predict that the higher order equation provides a statistically better fit.

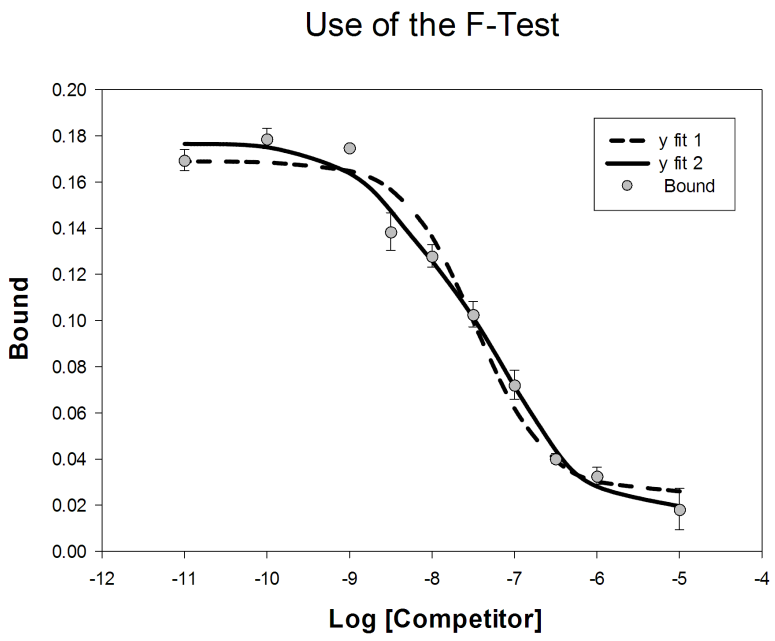


Figure 13: Comparing Two Curve Fits

R Squared for Nonlinear Regressions

You can use this transform to compute the coefficient of determination (R^2) for the results of a nonlinear regression. The original Y values and the Y data from the fitted curve are used to calculate R^2.

To save the fitted Y values of the nonlinear regression to the worksheet, use the Regression Wizard to save the Function results to the appropriate column (for this transform, column 3).

1. Place your original Y data in column 2 of the worksheet and the fitted Y data in column 3. If your data has been placed in other columns, you can specify these columns after you open the R2.XFM transform file. You can enter data into an existing or a new worksheet.
2. Click **Run**. The R^2 value is placed in column 4 of the worksheet, or in the column specified with the res variable.

Standard Deviation of Linear Regression Parameters

This transform computes linear 1st-order regression parameter values (slope and intercept) and their standard deviations using X and Y data sets of equal length.

To calculate 1st-order regression parameters and their standard deviations for XY data points:

1. Place the X data in column 1 of the worksheet and the Y data in column 2. If your data is in other columns, you can specify these columns after you open the STDV_REG.XFM transform file. You can enter data into an existing worksheet or a new worksheet.
2. Click **Run**. The results are placed in columns 3 and 4, or in the columns specified by the res variable.

Graphing Transform Examples

The graph transform examples are provided to show you how transform equations can manipulate and calculate data to create complex graphs.

Each of the following descriptions provide instructions on how to use SigmaPlot to create graphs. Most of these graphs, however, are already set up as sample graphs. If you use the provided worksheet and graphs with the corresponding transform files, SigmaPlot automatically creates the graphs after you run the transform.

Control Chart for Fractional Defectives with Unequal Sample Sizes

This example computes the fraction of defectives p for a set of unequally sized samples using their corresponding numbers of defects, the control limits for p, and data for the upper and lower control lines. This transform contains examples of the following transform functions:

- stddev
- sqrt

To calculate and graph the fraction of defectives and control lines for given sample sizes and number of defects per sample, you can either use the provided sample data and graph or begin a new notebook, enter your own data and create your own graph using the data.

1. **To use the provided sample data and graph,** open the Control Chart worksheet and graph in the Control Chart section of the Transform Examples notebook. The worksheet appears with data in columns 1, 2, and 3. The graph page appears with an empty graph.
2. **To use your own data,** place the sample sizes in column 1 and the corresponding number of defects data in column 2 of a new worksheet. If your data is in other columns, you can specify these columns after you open the CONTCHRT.XFM transform file. You can enter your data in an existing or a new worksheet.
3. Press **F10** to open the **User-Defined Transform** dialog box, then click **Open** to open the CONTCHRT.XFM transform file in the XFMS directory. The Control Chart transform appears in the edit window.
4. Click **Run**. The results are placed in columns 4 through 5 of the worksheet.
5. If you opened the Control Chart graph, view the graph page. The graph plots the fraction of defectives using a Line and Scatter plot with a Simple Straight Line style graphing column 3 as Y data versus the row numbers. The control lines are plotted as a Simple Horizontal Step Plot using columns 4 and 5 versus their row numbers. The mean line for the fractional defectives is drawn with a reference line.

6. **To create your own graph,** create a Line and Scatter Plot, with a Simple Line style, then plot column 3 as Y data against the row numbers. Add an additional Line Plot using the Multiple Horizontal Step Plot style, plotting columns 4 and 5 versus their two numbers, then add a reference line to plot the mean line for the fractional device.

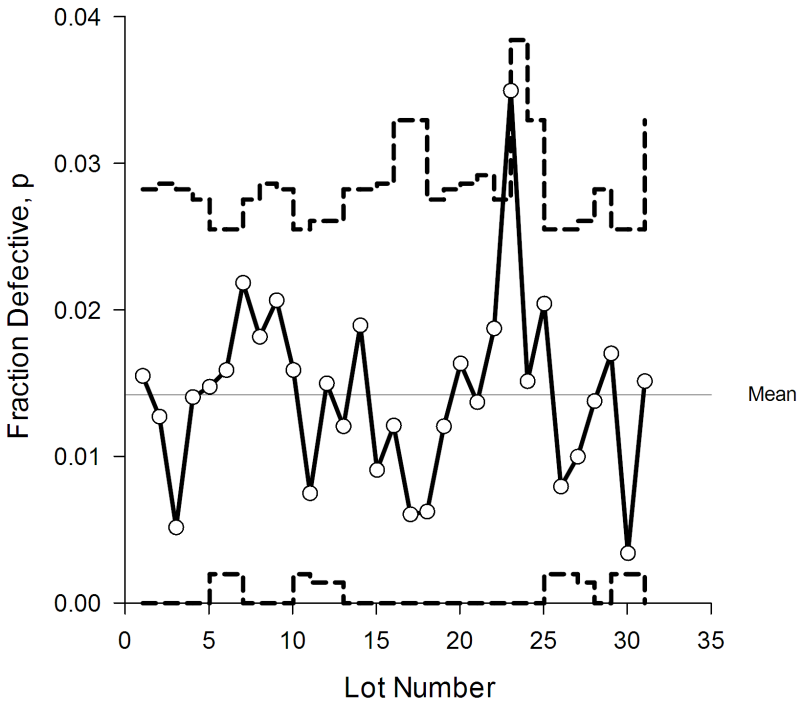


Figure 14: Control Chart Graph

Cubic Spline Interpolation and Computation of First and Second Derivatives

This example takes data with irregularly spaced X values and generates a cubic spline interpolant. The CBESPLN1.XFM transform takes X data which may be irregularly spaced and generates the coefficients for a cubic spline interpolant. The CBESPLN2.XFM transform takes the coefficients and generates the spline interpolant and its two derivatives.

The values for the interpolant start at a specified minimum X which may be less than, equal to, or greater than the X value of the original first data point. The interpolant has equally spaced X values that end at a specified maximum which may be less than, equal to, or greater than the largest X value of the original data.

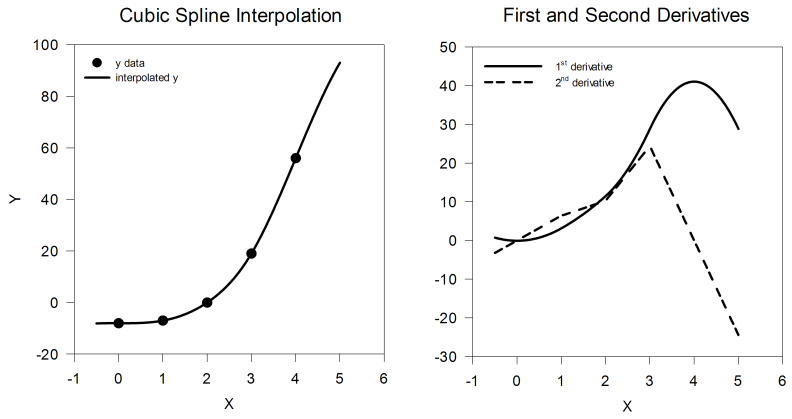
Note that this is not the same algorithm that SigmaPlot uses; this algorithm does not handle multiple valued functions, whereas SigmaPlot does.

To use the transform to generate and graph a cubic spline interpolant, you can either use the provided sample data and graph, or begin a new notebook, enter your own data and create your own graph using the data.

1. **To use the provided sample data and graph,** open the Cubic Spline worksheet and graph by double-clicking the graph page icon in the Cubic Spline section of the Transform Examples notebook. The worksheet appears with data in columns 1 and 2 and the graph page appears with two graphs. The first graph plots the original XY data as a scatter plot. The second graph appears empty.
2. **To use your own data,** enter the irregularly spaced XY data into the worksheet. The X values must be sorted in strictly increasing values. The default X and Y data columns used by the transform are columns 1 and 2, respectively.
3. Press **F10** to open the **User-Defined Transform** dialog box, then click **Open** to open the CBESPLN1.XFM transform file in the XFMS directory. The first Cubic Spline transform appears in the edit window.
4. Move to the Input Variables heading. Set the X data column variable cx, the Y data column cy, the beginning interpolated X value xbegin, the ending interpolated X value xend, and the X increments for the interpolated

points xstep. A larger X step results in a smoother curve but takes longer to compute. Enter the end condition setting iend for the interpolation.

5. Enter the end condition setting iend for the interpolation.
   - You can use first, second, or third order conditions.
   - If you have only a few data points, you should try different orders to see which one you like the most. See the example for the effect of too low an order on the first and second derivatives.
   - 1 end spline segments approach straight lines asymptotically
   - 2 end spline segments approach parabolas asymptotically
   - 3 end spline segments approach cubics asymptotically
6. Move to the RESULTS heading and enter the first column number for the results cr. This column for the beginning of the results block is specified in both transforms.
7. Click **Run** to run the transform. When it finishes, press F10 then open the CBESPLN2.XFM transform file in the XFMS directory. Make sure that the cr variable is identical to the previous value, then click Run.
8. If you opened the Cubic Spline graph, view the page. The first graph plots the original XY data as a scatter plot and the interpolated data as a second line plot by picking the cr column as the X column and cr+1 as the Y column. The second graph plots the derivatives as line plots using the cr column versus the cr+2 column and the cr column versus the cr+3 column.
9. **To create your own graphs using SigmaPlot,** create a Scatter Plot using a Simple Scatter style which plots the original data in columns 1 and 2 as XY pairs. Add an additional Line Plot using a Simple Spline Curve, then plot the cr column as the X column against the cr+1 column as the Y column.



Fast Fourier Transform

The Fast Fourier Transform converts data from the time domain to the frequency domain. It can be used to remove noise from, or smooth data using frequency-based filtering. Use the fft function to find the frequency domain representation of your data, then edit the results to remove any frequency which may adversely affect the original data.

The Fast Fourier Transform uses the following transform functions:

- fft
- invfft
- real
- img
- complex
- mulcpx
- invcpx

The Fast Fourier Transform operates on a range of real values or a block of complex values. For complex values there are two columns of data. The first column contains the real values and the second column represents the imaginary values. The worksheet format of a block of complex numbers is:

r_1	i_1
r_2	i_2
....	....
r_n	i_n

where r values are real elements, and i values are imaginary elements. In transform language syntax, the two columns $\{\{r_1, r_2, ... r_n\},\{i_1, i_2, ... i_n\}\}$ are written as:

block($\{r_1, r_2, ... r_n\},\{i_1, i_2, ... i_n\}$)

This function works on data sizes of size 2^n numbers. If your data set is not 2^n in length, the fft function pads 0 at the beginning and end of the data range to make the length 2^n.

For more information, see Low Pass Filter on page 47.

The fft function returns a range of complex numbers. The Fast Fourier Transform is usually graphed with respect to frequency. To produce a frequency scale, use the relationship:

```
f=fs*(data(0,n/2)-1)/n
```

where *fs* is the sampling frequency. The example transform POWSPEC.XFM. includes the automatic generation of a frequency scale.

The Fast Fourier Transform operates on data which is assumed to be periodic over the interval being analyzed. If the data is not periodic, then unwanted high frequency components are introduced. To prevent these high frequency components from occurring, windows can be applied to the data before using the fft transform. The Hanning window is a cosine function that drops to zero at each end of the data. The example transform POWSPEC.XFM includes the option to implement the Hanning window. For more information, see Computing Power Spectral Density on page 35.

Computing Power Spectral Density

The example transform POWSPEC.XFM uses the Fast Fourier Transform function, then computes the power spectral density, a frequency axis, and makes optional use of a Hanning window.

To calculate and graph the power spectral density of a set of data, you can either use the provided sample data and graph, or begin a new notebook, enter your own data and create your own graph using the data.

1. **To use the sample worksheet and graph,** open the Power Spectral Density worksheet and graph by double-clicking the graph page icon in the Power Spectral Density section of the Transform Examples notebook. Data appears in column 1 of the worksheet, and two graphs appear on the graph page. The top graph shows data generated by the sum of two sine waves plus Gaussian random noise. The data is represented by:

   ```
   f(t)=sin(2*pi*f1*t)+0.3*sin(2*pi*f2*t)+g(t)
   ```

   where `f1=10 cycles/sec (cps), f2=100cps`, and the Gaussian random noise has mean 0 and standard deviation of 0.2. The lower graph is empty.
2. **To use your own data,** place your data in column 1. If your data is in a different column, specify the new column after you open the POWSPEC.XFM transform file.
3. Press **F10** to open the **User-Defined Transform** dialog box, then click **Open** to open POWSPEC.XFM transform file in the XFMS directory. The Power Spectral Density transform appears in the edit window.

   Note: To use this transform, the Trigonometric Units must be set to Radians.
4. Click **Run**.

   Since the frequency sampling value (fs) is nonzero, a frequency axis is generated in column 2 and the power spectral density data in column 3.
5. If you opened the Power Spectral Density graph, view the graph page. Two graphs appear on the page. The top graph plots the data generated by the sum of two sine waves plus Gaussian random noise using a Line Plot with

Simple Straight Line style graphing column 1 versus row numbers. The lower graph plots the power spectral density using a Line Plot with a Simple Straight Line style, graphing column 2 as the X data (frequency), and column 3 as the Y data.

6. **To plot your own data using SigmaPlot,** click the **Create Graph Graph** tab . Create a Line Plot with a Simple Straight Line style plotting your original data versus row numbers by choosing Single Y data format. If you set the frequency sampling value (fs) to nonzero, create a Line Plot with a Simple Straight Line style, graphing columns 2 and 3 using XY Pair data format. Otherwise, create a Line Plot with a Simple Straight Line style plotting column 3 (power spectral density) versus row numbers by choosing Single Y data format.

   The power spectral density plot of the signal *f(t)* shows two major peaks at the two frequencies of the sine waves (10cps and 100cps), and a more or less constant noise level in between.

   The top graph shows *f(t)* data generated by the sum of two sine waves plus Gaussian random noise. The bottom graph is the power spectral density of the signal *f(t)*.

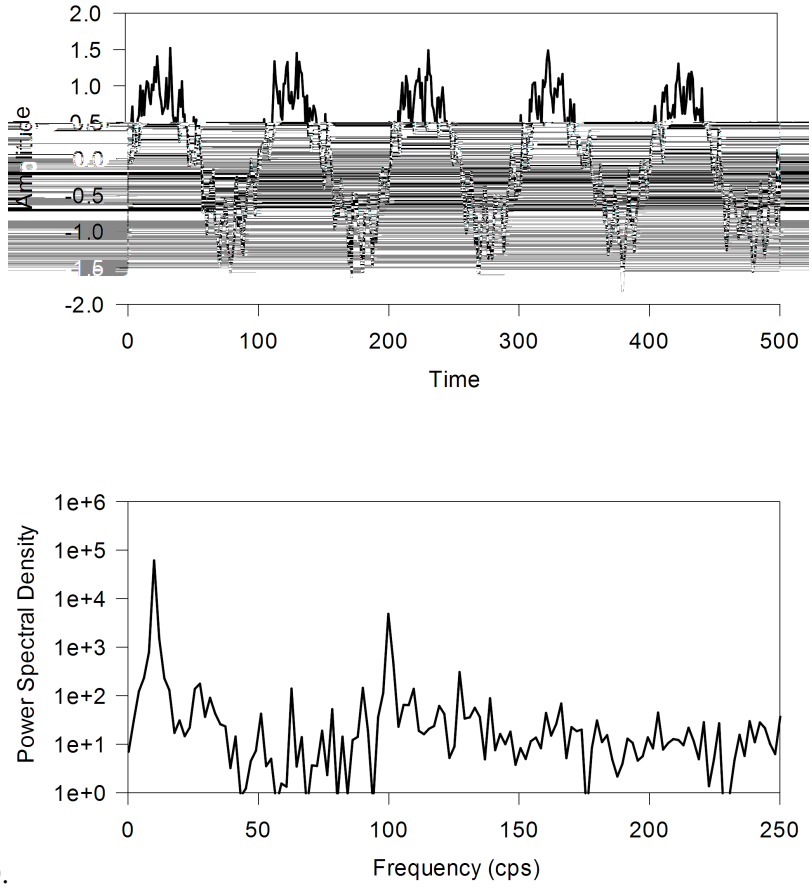


Figure 15: Power Spectral Density Example Graph

Using the Block Function

To return the full fft data to the worksheet:

1. First assign the data you want to filter to column 1 of the worksheet. You can generate the data using a transform, or use your own measurements.
2. Press **F10** to open the **User-Defined Transforms** dialog box, then click **New** to start a new transform.
3. Type the following transform in the edit window:

```
x=col(1)     'real data
tx=fft(x)    'compute the fft
block(2)=tx  'place real fft data back in col(2)
```

```
        'place imaginary fft data in col(3)
```

4. Click **Run**. The results are placed starting one column over from the original data.

Kernel Smoothing

The example transform SMOOTH.XFM smooths data by convolving the Fast Fourier Transform of a triangular smoothing kernel together with the fft of the data. Smoothing data using this transform is computationally very fast; the number of operations is greatly reduced over traditional methods, and the results are comparable. To increase the smoothing, increase the width of the triangular smoothing kernel.

To calculate and graph the smoothed data, you can either use the provided sample data and graph, or begin a new notebook, enter your own data, and create your own graph using the data.

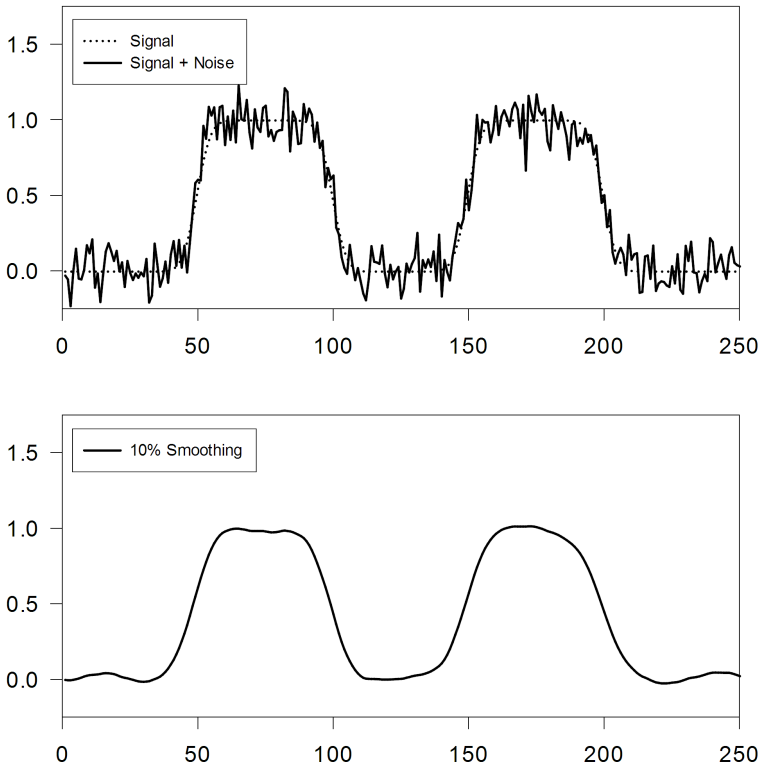
1. **To use the sample worksheet and graph,** open the Kernel Smoothing worksheet and graph by double-clicking the graph page icon in the Kernel Smoothing section of the Transform Examples notebook. Data appears in columns 1 through 4, 6, and 7 of the worksheet, and two graphs appear on the graph page. The first graph has two plots, the signal, and the signal with noise distortion. Column 1 contains the X data, column 2 contains the Y data for the signal, and column 3 contains the Y data for the signal and the noise distortion. The lower graph is empty.
2. **To use your own data,** place your data in columns 1 through 2. If your data is in other columns, specify the new columns after you open the SMOOTH.XFM transform file. If necessary, specify a new column for the results.
3. Press **F10** to open the **User-Defined Transform** dialog box, then click **Open** to open SMOOTH.XFM transform file in the XFMS directory. The Kernel Smoothing transform appears in the edit window.

   Note: To use this transform, make sure the Insert mode is turned off.

4. Click **Run**. The results are placed in column 5 unless you specified a different column in the transform.
5. If you opened the Kernel Smoothing graph, view the graph page. Two graphs appear on the page. The first graph has two plots, the signal, and the signal with noise distortion. The Line Plot with a Multiple Straight Line style graphs column 1 as the X data, column 2 as the Y data for the signal, and column 3 as the Y data for the signal and the noise distortion. The lower Line Plot with a Simple Straight Line style plots column 1 as the X data, and column 5 as the Y data using XY Pairs data format.
6. **To plot your own data using SigmaPlot,** click the **Create Graph Graph** tab . Create a Line Plot with a Multiple Straight Line style using X Many Y data format, plotting column 1 as the X data, column 2 as the Y data for the signal, and column 3 as the Y data for the signal and the noise distortion. Create a second Line Plot graph with a

Simple Straight Line style using the data in columns 1 and 5, graphing column 1 as the X data and column 5 as the Y data using XY Pairs data format.

The top graph shows two plots: the signal, and the signal plus noise distortion.
The bottom graph is the kernel smoothing of the signal with smoothing set at



10%.

Figure 16: Kernel Smoothing Graph

Smoothing with a Low Pass Filter

The Low Pass Filter transform smooths data by eliminating high frequencies. Use this transform in contrast to the Kernel Smoothing transform which smooths data by augmenting some frequencies while minimizing others. The transform statements describing how the low pass filter works are:

```
x=col(1)     'the data to smooth
f=5        'number of channels to eliminate

tx=fft(x)        'fft of data
r=data(1,size(tx)/2) 'total number of channels
mp=size(tx)/4      'get the midpoint
           'remove the frequencies
td=if( r<mp-f or r>mp+1+f,tx,0)
sd=invfft( td )     'convert back to time domain

col(2)=real(sd)     'save smoothed data to worksheet
```

The LOWPASS.XFM transform expresses *f* as a percentage for ease of use. As the value of *f* increases, more high frequency channels are removed. Note that this is a digital transform which cuts data at a discrete boundary. In

addition, this transform does not alter the phase of the data, which makes it more accurate than analog filtering. A high pass or band pass filter can be constructed in the same manner.

To calculate and graph the smoothing of a set of data using a low pass filter, you can either use the provided sample data and graph, or begin a new notebook, enter your own data, and create your own graph using the data.

1. **To use the sample worksheet and graph,** open the Low Pass Smoothing worksheet and graph by double-clicking the graph page icon in the Low Pass Smoothing section of the Transform Examples notebook. Data appears in columns 1 through 4 of the worksheet, and two graphs showing plots appear on the graph page. Column 1 contains the X data, column 2 contains the Y data for the signal and the noise distortion, column 3 contains the X data, and column 4 contains the Y data for the original signal. The top graph plots the signal plus the noise distortion; the bottom graph plots the signal.
2. **To use your own data,** place your data in columns 1 through 2. If your data is in other columns, specify the new columns after you open the LOWPASS.XFM transform file. If necessary, specify a new column for the results.
3. Press **F10** to open the **User-Defined Transform** dialog box, then click **Open** to open LOWPASS.XFM transform file in the XFMS directory. The Low Pass Filter transform appears in the edit window.

   Note: To use this transform, make sure Insert mode is turned off.

4. Click **Run**. The results are placed starting in column 5, unless you specified a different column in the transform.
5. If you opened the Low Pass Smoothing graph, view the graph page. Two graphs appear. The top graph plots the signal plus the noise distortion, using a Line Plot with a Simple Straight Line style and XY Pairs data format graphing column 1 as the X data, column 2 as the Y data for the signal and the noise distortion. The bottom graph displays two plots. A Scatter Plot with a Simple Scatter Style and XY Pairs data format, plots column 3 as the X data, and column 4 as the Y data for the original signal. A second Line Plot with a Simple Straight Line style using data in columns 1 and 5, plots column 1 as the X data and column 5 as the Y data using XY Pairs data format.
6. **To plot your own data using SigmaPlot,** click the **Create Graph Graph** tab . Graph the signal plus the noise distortion, using a Line Plot with a Simple Straight Line style and XY Pairs data format graphing column 1 as the X data, column 2 as the Y data for the signal and the noise distortion. Create a second graph with two plots. Plot the original signal using a Scatter Plot with a Simple Scatter Style and XY Pairs data format, plotting column 3 as the X data, and column 4 as the Y data for the original signal. Add a second Line Plot with a Simple Straight Line

style using data in columns 1 and 5, plotting column 1 as the X data and column 5 as the Y data using XY Pairs data format.

The top graph shows the signal plus noise distortion. The bottom graph shows the signal and the low pass filtering set at 88%.

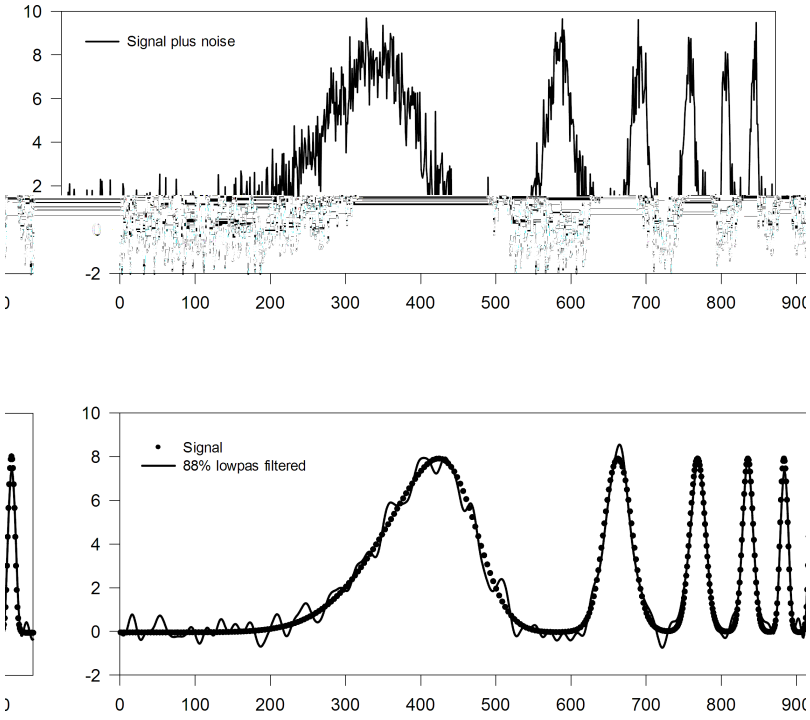


Figure 17: Low Pass Filter Smoothing Graph

Gain Filter Smoothing

The GAINFILT.XFM transform example demonstrates gain filter smoothing. This method eliminates all frequencies with power spectral density levels below a specified threshold. The transform statements describing how gain filter smoothing works are:

```
P=4000        'psd threshold
x=col(1)       'data

tx=fft(x)           'compute fft of data
md=real(tx)^2+img(tx)^2   'compute sd
kc=if(md>P,1,0)       'remove frequencies with
              'psd<P
sd=mulcpx(complex(kc),tx)  'remove frequency
              'components from x
td=real( invfft(sd) )    'convert back to time domain
col(2)=td          'place results in worksheet
```

To calculate and graph the smoothing of a set of data using a gain filter, you can either use the provided sample data and graph, or begin a new notebook, enter your own data, and create your own graph using the data.

1. **To use the sample worksheet and graph,** open the Gain Filter Smoothing worksheet and graph by double-clicking the graph page icon in the Gain Filter Smoothing section of the Transform Examples notebook. Data appears in columns 1 through 3 of the worksheet, and two graphs showing plots, and one blank graph appear on the graph page. Column 1 contains the Y data for the signal plus noise, column 2 contains the X data and column 3 contains the Y data for the power spectral density graph. The top graph plots the signal plus the noise distortion; the middle graph plots the power spectral density.

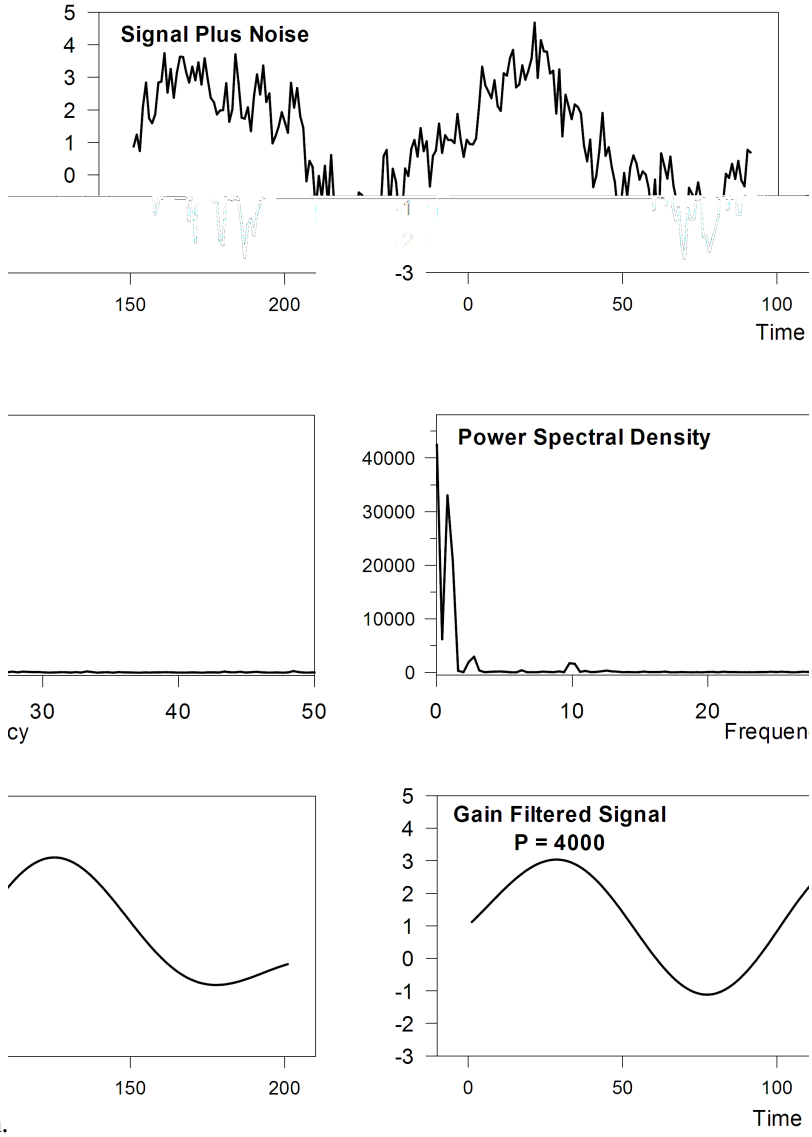
2. **To use your own data,** place your data in column 1. If your data is in a different column, specify the new column after you open the GAINFILT.XFM transform file. If necessary, specify a new column for the results.
3. Press **F10** to open the **User-Defined Transform** dialog box, then click **Open** to open GAINFILT.XFM transform file in the XFMS directory. The Gain Filter transform appears in the edit window.

   Note: To use this transform, make sure Insert mode is turned off.

4. Click **Run**. The results are placed in column 5 unless you specified a different column in the transform.
5. If you opened the Gain Filter Smoothing graph, view the graph page. Three graphs appear. The top graph plots the signal plus the noise distortion using a Line Plot with a Simple Straight line style and Single Y data format, plotting column 1 as the Y data for the signal plus noise. The middle graph plots the power spectral density using a Line Plot with a Simple Straight Line style and XY Pairs data format, plotting column 2 as the X data and column 3 as the Y data for the power spectral density graph. The lower graph is a plot of the gain filtered signal, using a Line Plot with a Simple Straight Line style, and single Y data format from column 5.
6. **To plot your own data using SigmaPlot,** click the **Create Graph Graph** tab . Create two graphs. Plot the signal plus the noise distortion using a Line Plot with a Simple Straight line style and Single Y data format, plotting

column 1 as the Y data for the signal plus noise. Plot the gain filtered signal using a Line Plot with a Simple Straight Line style, and single Y data format from column 5.

The top graph shows the signal plus noise distortion. The middle graph shows the power spectral density of the signal plus noise distortion. The lower graph shows the gain filter smoothed



data.

Figure 18: Gain Filter Smoothing Graph

Frequency Plot

This transform example creates a frequency plot showing the frequency of the occurrence of data in the Y direction. Data is grouped in specified intervals, then horizontally plotted for a specific Y value. Parameters can be set to display symbols that are displaced a specific distance from each other or that touch or overlap. You can also plot the mean value of each data interval. This transform example shows overlapping symbols which give the impression of data mass.

To calculate and graph the frequency of the occurrence of a set of data, you can either use the provided sample data and graph, or begin a new notebook, enter your own data and create your own graph using the data.

1. **To use the sample worksheet and graph,** open the Frequency Plot worksheet and graph by double-clicking the graph page icon in the Frequency Plot section of the Transform Examples notebook. Data appears in columns 1 through 3 of the worksheet, and an empty graph appears on the graph page.

2. **To use your own data**, place your data in columns 1 through 3. You can put data in as many or as few columns as desired, but if you use the sample transform you must change the X locations of the Y values in the second line under the Input heading in the transform file to reflect the number of data columns you are using. If your data is in other columns or more than three columns, specify the new columns after you open the FREQPLOT.XFM transform file.

   Enter the tick labels for the X axis in a separate column.

3. **To create your own graph using SigmaPlot,** make a graph with three Scatter Plots with Simple Scatter styles. Plot each consecutive result column pair as XY pair scatter plots. If the mean line option is active in the transform, plot the last consecutive result column pair as a XY pair Line Plot with Simple Straight Line style. Use labels typed into a worksheet column as the X axis tick labels.
4. Press **F10** to open the **User-Defined Transform** dialog box, then click **Open** to open the FREQPLOT.XFM transform file in the XFMS directory. The Frequency Plot transform appears in the edit window.
5. Click **Run**. The results are placed starting one column over from the original data.
6. If you opened the sample Frequency Plot graph, view the graph page. A Scatter Plot appears plotting columns 5 and 6, 7 and 8, and 9 and 10 as three separate XY Pair plots. The lines passing through each data interval is a fourth Line Plot with a Simple Straight Line style plotting columns 11 and 12 as an XY pair, representing the mean value of each data interval. The X axis tick marks are generated by the transform. The axis labels are taken from column 13.

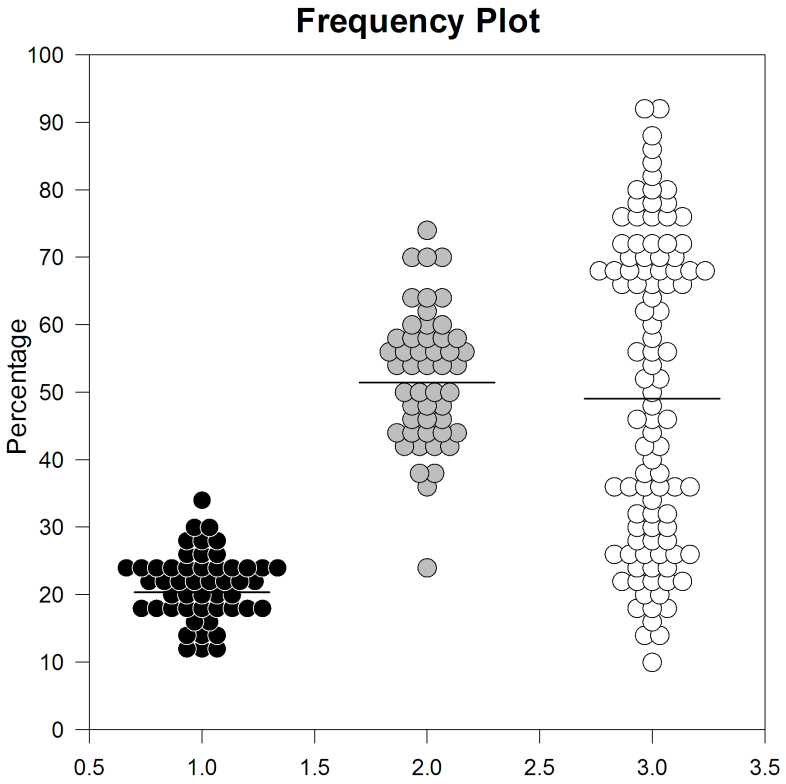


Figure 19: Frequency Plot Graph

7. **To create your own graph using SigmaPlot,** make a graph with three Scatter Plots with Simple Scatter styles. Plot each consecutive result column pair as XY pair scatter plots. If the mean line option is active in the transform, plot the last consecutive result column pair as a XY pair Line Plot with Simple Straight Line style. Use labels typed into a worksheet column as the X axis tick labels.

Gaussian Cumulative Distribution from the Error Function

Rational approximations can be used to compute many special functions. This transform demonstrates a polynomial approximation for the error function. The error function is then used to generate the Gaussian cumulative distribution

function. The absolute maximum error for the error function approximation is less than 2.5 x 10-5 (M. Abramowitz and L.A. Stegun, Handbook of Mathematical Functions, p. 299).

To calculate and graph the Gaussian cumulative distribution for given X values, you can either use the provided sample data and graph or begin a new notebook, enter your own data and create your own graph using the data.

1. **To use the sample worksheet and graph,** open the Gaussian worksheet and graph by double-clicking the graph page icon in the Gaussian section of the Transform Examples notebook. Data appears in column 1 of the worksheet and two empty graphs appear on the graph page.
2. **To use your own data,** place the X data in column 1. If your data has been placed in another column, you can specify the column after you open the GAUSDIST.XFM transform file.
3. Press **F10** to open the **User-Defined Transform** dialog box, then click **Open** to open the GAUSDIST.XFM transform file in the XFMS directory. The Gaussian Cumulative transform appears in the edit window.
4. Click **Run**. The results are placed in column 2, or in the column specified by the res variable.
5. If you opened the sample Gaussian graph, view the graph page. A Line Plot appears with a spline curve in the first graph with column 1 as the X data versus column 2 as the distribution (Y) data.
6. **To create your own graph using SigmaPlot,** make a Line Plot graph with a Simple Spline Curve. The spline curve plots column 1 as the X data versus column 2 as the distribution (Y) data.

Gaussian Cumulative Distribution on a Probability Scale

The probability scale is the inverse of the Gaussian cumulative distribution function. When a Gaussian cumulative distribution function is graphed using the probability scale, the result is a straight line.

1. If you opened the sample Gaussian graph, view the graph page. A straight line plot appears in the second graph plotting the distribution data in column 3 along a probability scale.
2. **To create your own graph using SigmaPlot,** create a Line Plot with a Simple Straight Line using column 1 as your X data and column 3 as your Y data, and set the Y axis scale to Probability.

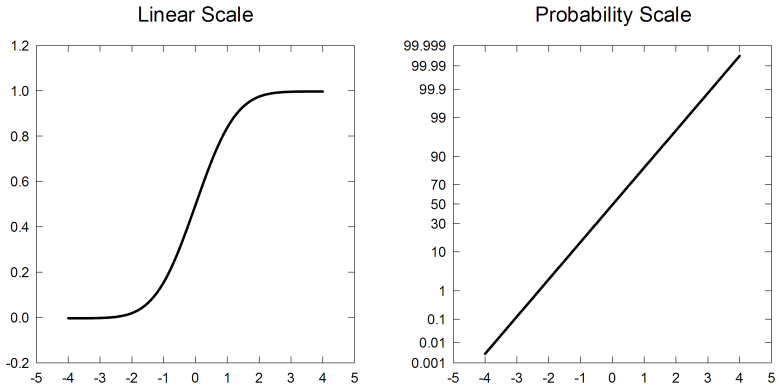


Figure 20: Gaussian Cumulative Distribution Graphs

Histogram with Gaussian Distribution

This transform calculates histogram data for a normally distributed sample, then uses the sample mean and standard deviation of the histogram to compute and graph a Gaussian distribution for the histogram data.

The Histogram Gaussian transform uses examples of the following functions:

- Gaussian
- histogram
- size
- [...] (array reference)

To calculate and graph a histogram and Gaussian curve for a normally distributed sample, you can either use the provided sample data and graph or begin a new notebook, enter your own data, and create your own graph using the data.

To use the sample worksheet and graph:

1. Open the Histogram Gaussian worksheet and graph by double-clicking the graph page icon in the Histogram Gaussian section of the Transform Examples notebook.

   The Histogram worksheet with data in column 1 and an empty graph page appears. The data in the Histogram Gaussian worksheet was generated using the transform:

   ```
   col(1) = gaussian(100,0,325,2)
   ```

2. Place the sample in column 1 of the worksheet. If your data has been placed in another column, you can specify this column after you open the HISTGAUS.XFM transform file. You can enter the data into an existing or new worksheet.
3. Press **F10** to open the **User-Defined Transform** dialog box, then click **Open** to open the HISTGAUS.XFM transform file in the XFMS directory. The Histogram with Gaussian Distribution transform appears in the edit window.
4. Click **Run**. The results are placed in columns 2 through 5 of the worksheet, or in the columns specified by the res variable.
5. If you opened the Histogram Gaussian graph, view the graph page. A histogram appears using column 2 as X data versus column 3 as the Y data. The curve plots the Gaussian distribution using column 4 as X data versus column 5 as the Y data.

6. **To create your own graph using SigmaPlot,** create a simple vertical bar chart and set the bar widths as wide as possible. Add the Gaussian curve to the graph by creating another plot using the data in column 4 as the X data and the data in column 5 as the Y data.

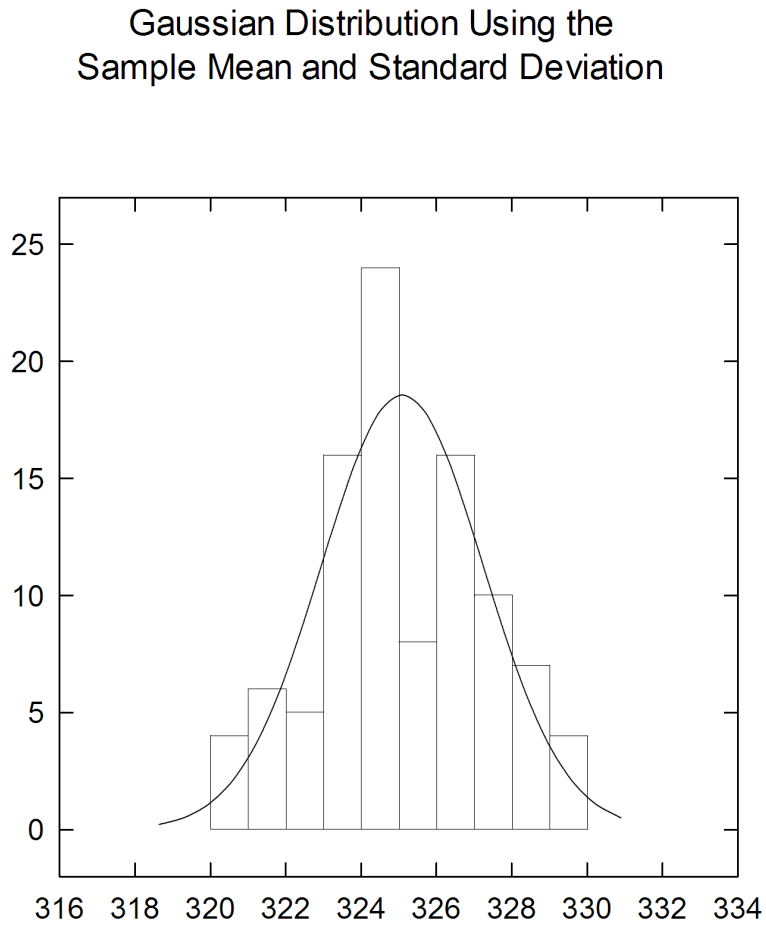


Figure 21: The Histogram Gaussian Graph

Linear Regression with Confidence and Prediction Intervals

This transform computes the linear regression and upper and lower confidence and prediction limits for X and Y columns of equal length. A rational polynomial approximation is used to compute the t values used for these confidence limits.

The figure below displays the sample Linear Regression graph with the results of the LINREGR.XFM transform plotted.

The LINREGR.XFM transform contains examples of these two functions:

- min
- max

To calculate and graph a linear regression and confidence and prediction limits for XY data points, you can either use the provided sample data and graph or begin a new notebook, enter your own data, and create your own graph using the data.

1. **To use the provided sample data and graph,** open the Linear Regression worksheet and graph by double-clicking the graph page icon in the Linear Regression section of the Transform Examples notebook. The worksheet appears with data in columns 1 and 2. The graph page appears with a scatter graph plotting the original data in columns 1 and 2.

2. **To use your own data,** place the X data in column 1 and the Y data in column 2. If your data has been placed in other columns, you can specify these columns after you open the LINREGR.XFM transform file. You can enter data into an existing or a new worksheet.
3. Press **F10** to open the **User-Defined Transform** dialog box, then click **Open** to open the LINREGR.XFM transform in the XFMS directory. The Linear Regression transform appears in the edit window. If necessary, change the x_col, y_col, and res variables to the correct column numbers (this is not necessary for the example Linear Regression worksheet data).
4. Change the Z variable to reflect the desired confidence level (this is not necessary for the example Linear Regression worksheet data).
5. Click **Run**. The results are placed in columns 3 through 8, or in the columns specified by the res variable.
6. If you opened the Linear Regression graph, view the graph page. The original data in columns 1 and 2 is plotted as a scatter plot. The regression is plotted as a solid line plot using column 3 as the X data versus column 4 as the Y data, the confidence limits are plotted as dashed lines using column 3 as a single X column versus columns 7 and 8 as many Y columns, and the prediction limits are plotted as dotted lines using column 3 as a single X column versus columns 7 and 8 as many Y columns.
7. **To create your own graph in SigmaPlot,** create a Scatter Plot with a Simple Regression, plotting column 1 against column 2 as the symbols and using column 3 plotted against column 4 as the regression. Add confidence and prediction intervals using column 3 as the X column and columns 7 and 8 as the Y columns.

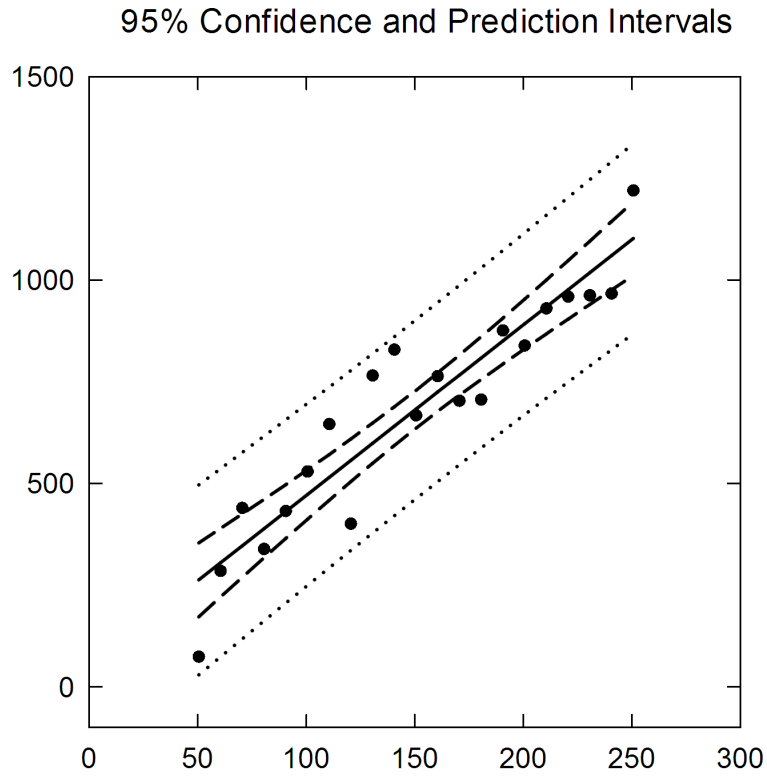


Figure 22: Linear Regression Graph

Low Pass Filter

This transform is a smoothing filter which produces a data sequence with reduced high frequency components. The resulting data can be graphed using the original X data.

To calculate and graph a data sequence with reduced high frequency components, you can either use the provided sample data and graph or begin a new notebook, enter your own data, and create your own graph using the data.

1. **To use the provided sample data and graph,** double-click the Low Pass Filter graph page icon in the Low Pass Filter section of the Transform Examples notebook. The worksheet appears with data in columns 1 and 2. The

graph page appears with two graphs. The first is a line graph plotting the raw data in columns 1 and 2. The second graph is empty.

2. **To use your own data,** place your Y data (amplitude) in column 2 of the worksheet, and the X data (time) in column 1. If your data is in other columns, you can specify these columns after you open the LOWPFILT.XFM file. You can enter your data in an existing or new worksheet.
3. Press **F10** to open the **User-Defined Transform** dialog box, then click **Open** to open the LOWPFILT.XFM transform file in the XFMS directory. The Low Pass Filter transform appears in the edit window.
4. Set the sampling interval dt (the time interval between data points) and the half power point fc values. The half power point is the frequency at which the squared magnitude of the frequency response is reduced by half of its magnitude at zero frequency.
5. If necessary, change the cy1 source column value and cy2 filtered data results to the correct column numbers.
6. Click **Run** to run the transform. Filtered data appears in column 3 in the worksheet, or in the worksheet column you specified in the transform.
7. If you opened the Low Pass Filter graph, view the graph page. The second graph appears as a line graph plotting the smoothed data in columns 1 and 3.
8. **To create your own graphs in SigmaPlot,** create the first graph as a Line Plot with a Simple Spline Curve using the raw data in columns 1 and 2 as the X and Y data. Make the second Line Plot graph with a Simple Spline Curve using the data in column 1 as the X data and the smoothed data in column 3 as the Y data.

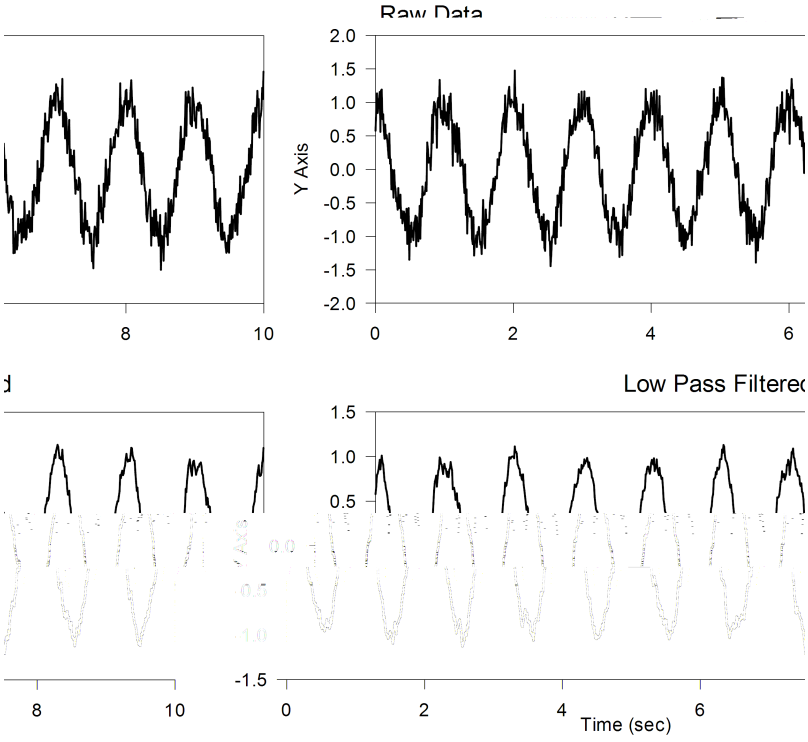


Figure 23: Low Pass Filter Graph Plotting Raw Data and Filtered Data

Lowess Smoothing

Smoothing is used to elicit trends from noisy data. Lowess smoothing produces smooth curves under a variety of conditions. "Lowess" means locally weighted regression. Each point along the smooth curve is obtained from a regression of data points close to the curve point with the closest points more heavily weighted.

The y value of the data point is replaced by the y value on the regression line. The amount of smoothing, which affects the number of points in the regression, is specified by the user with the parameter *f*. This parameter is the fraction of the total number of points that is used in each regression. If there are 50 points along the smooth curve with $f = 0.2$ then 50 weighted regressions are performed and each regression is performed using 10 points.

An example of the use of lowess smoothing for the U.S. wheat production from 1872 to 1958 is shown in the figures below. The smoothing parameter *f* was chosen to be 0.2 since this produced a good trade-off between noisy under-smoothing and over-smoothing which misses some of the peak-and-valley details in the data.

1. **To use the provided sample data and graph,** open the Lowess Smoothing worksheet and graph in the Lowess Smoothing section of the Transform Examples notebook. The worksheet appears with data in columns 1, 2, and 3.
2. **To use your own data,** enter the XY data for your curve in columns 1 and 2, respectively. If your data has been placed in other columns, you can specify these columns after you open the LOWESS.XFM transform file. Enter data into an existing or a new worksheet.
3. Press **F10** to open the **User-Defined Transform** dialog box, then click **Open** to open the LOWESS.XFM transform file in the Transforms directory. The Lowess transform appears in the edit window.
4. Click **Run** . The results are placed in column 3 of the worksheet, or in the column specified by the ouput variable.
5. If you opened the Lowess Smoothing graph, view the graph page. The smoothed curve is plotted on the second graph and both the original and smoothed data are plotted on the third.

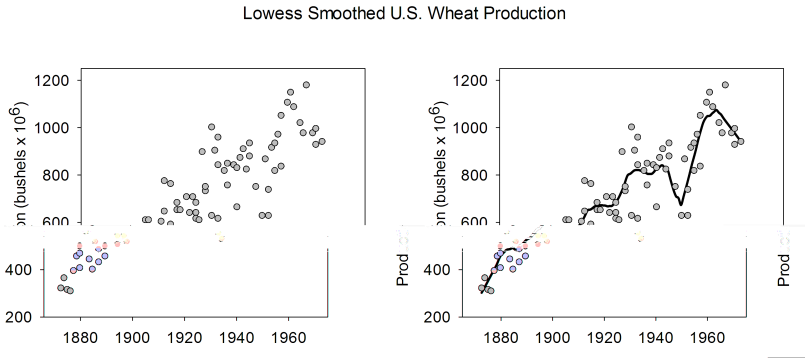


Figure 24: U.S. Wheat data and the lowess smoothed curve (f = 0.2). Notice the definite decreased production during World War II.

If you want to plot your own results, create a line plot of column 1 versus column 3.

Normalized Histogram

This simple transform creates a histogram normalized to unit area. The resulting data can be graphed as a bar chart. Histogram bar locations are shifted to be placed over the histogram box locations. The resulting bar chart is an approximation to a probability density function.

To calculate and graph a normalized histogram sample, you can either use the provided sample data and graph or begin a new notebook, enter your own data, and create your own graph using the data.

1. To use the provided sample data and graph, open the **Normalized Histogram** worksheet and graph in the **Normalized Histogram** and **Graph** section of the **Transform Examples** notebook. The worksheet appears with data in column 1. The data is made up of exponentially distributed random numbers generated with the transform:

```
x = random(200,1,1.e-10,1)col(1) = -ln(x)
```

   The graph page appears with an empty graph.
2. **To use your own data,** place your data in column 1 of the worksheet. If your data has been placed in another column, you can specify this column after you open the NORMHIST.XFM transform file. You can enter data into an existing or new worksheet.
3. Press **F10** to open the **User-Defined Transform** dialog box, then click **Open** to open the NORMHIST.XFM transform file in the XFMS directory. The Normalized Histogram transform appears in the edit window.
4. Click **Run**. The results are placed in columns 2 and 3 of the worksheet, or in the columns specified by the res variable.
5. If you opened the Normalized Histogram graph, view the graph page. A histogram appears using column 2 as X data versus column 3 as the Y data.

6. **To create your own graph in SigmaPlot,** create a Vertical Bar chart with simple bars, then set the bar widths as wide as possible.

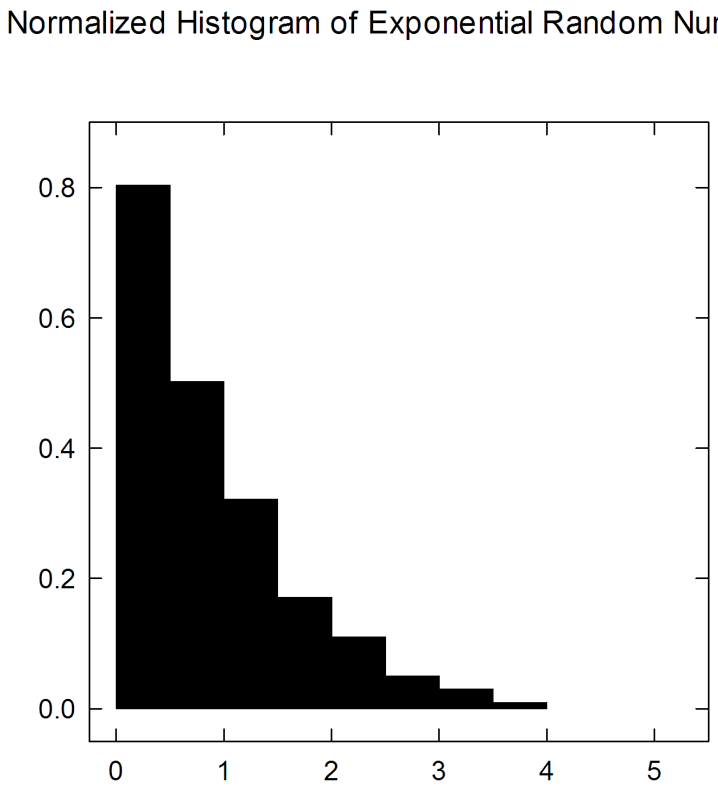


Figure 25: Normalized Histogram Graph

Smooth Color Transition Transform

This transform example creates a smooth color transition corresponding to the changes across a range of values. The transform places color cells in a worksheet column that change from a specified start color to a specified end color, each color cell incrementing an equivalent shade for each data value in the range. This transform example shows how the color transform can be set to display a "cool" (blue) color that corresponds to small residuals, and a "hot" (red) color that corresponds to large residuals resulting from a nonlinear regression. Since residuals vary positively and negatively about zero, the absolute values for the residuals are used in the transform.

Tip: It is unnecessary to sort the data before executing the smooth color transition transform.

To calculate and graph the smooth color transition of a set of data, you can either use the provided sample data and graph, or begin a new notebook, enter your own data, and create your own graph using the data.

1. **To use the sample worksheet and graph,** open the **Smooth Color Transition** worksheet and graph by double-clicking the graph page icon in the **Smooth Color Transition** section of the **Transform Examples** notebook. Data appears in columns 1 and 2 of the worksheet, and a scatter graph appears on the graph page.
2. **To use your own data,** place your data in columns 1 and 2. For the residuals example, column 2 is the absolute value of the residuals in column 1. To obtain absolute values of your data, use the `abs` transform function. For example, to obtain the absolute values of the data set in column 1, type the following transform in the User-Defined Transform dialog box:

```
col(2)=abs(col(1))
```

3. If your data is in a different column, specify the new column after you open the RGBCOLOR.XFM transform file.
4. **To create your own graph using SigmaPlot,** make a Scatter Plot graph with a Scatter Plot with Simple Scatter style. Plot the data as Single Y data format. Use the color cells produced by the transform by selecting the corresponding worksheet column from the **Symbol Fill Color** drop-down list.

Survival (Kaplan-Meier) Curves with Censored Data

This transform creates Kaplan-Meier survival curves with or without censored data. The survival curve may be graphed alone or with the data.

To use the transform, you can either use the provided sample data and graph or begin a new notebook, enter your own data, and create your own graph using the data.

1. **To use the sample worksheet and graph,** double-click the graph page icon in the Survival section of the Transforms Examples notebook. The Survival worksheet appears with data in columns 1 and 2. The graph page appears with an empty graph.
2. **To use your own data,** enter survival times in column 1 of the worksheet. Ties (identical survival times) are allowed. You can enter data into an existing or a new worksheet.
3. Enter the censoring identifier in column 2. This identifier should be 1 if the corresponding data point in column 1 is a true response, and 0 if the data is censored.
4. If desired, save the unsorted data by copying the data to two other columns.
5. Select columns 1 and 2, then on the **Worksheet** tab click **Sort Selection**. Specify the key column in the **Sort Selection** dialog box as column 1, and the sort order option as **Ascending**.
6. Check for any ties between true response and censored data. If any exist, make sure that within the tied data, the censored data follows the true response data.
7. Click **Run** to run the file. The sorted time, cumulative survival probability, and the standard error are placed in columns `res`, `res+1`, and `res+2`, respectively. For graphical purposes a zero, one, and zero have been placed in the first rows of the sorted time, cumulative survival curve probability and standard error columns.
8. If you opened the sample Survival graph, view the page. The Simple Horizontal Step Plot graphs the survival curve data from columns `res` as the X data versus column `res+1` as the Y data and a Scatter Plot graphs the data from the same columns. The first data point of the Scatter Plot at (0,1) is not displayed by selecting rows 2 to end in the Portions of Columns Plotted area of the Data section in the Plots tab of the Graph Properties dialog box. As shown in the figure below, a tied censored data point has been incorrectly placed; it should follow uncensored data.
9. **To graph a survival curve,** create a Line graph with a Simple Horizontal Step Plot graphing column res as the X data versus column res+1 as the Y data. If desired, create an additional Scatter plot, superimposing the survival data using the same columns for X data and Y data. To turn off the symbol drawn at x = 0 and y = 1, select Plot 2 and set Only rows = 2 to end by 1 in the Plots tab and Data sections of the Graph Properties dialog box.

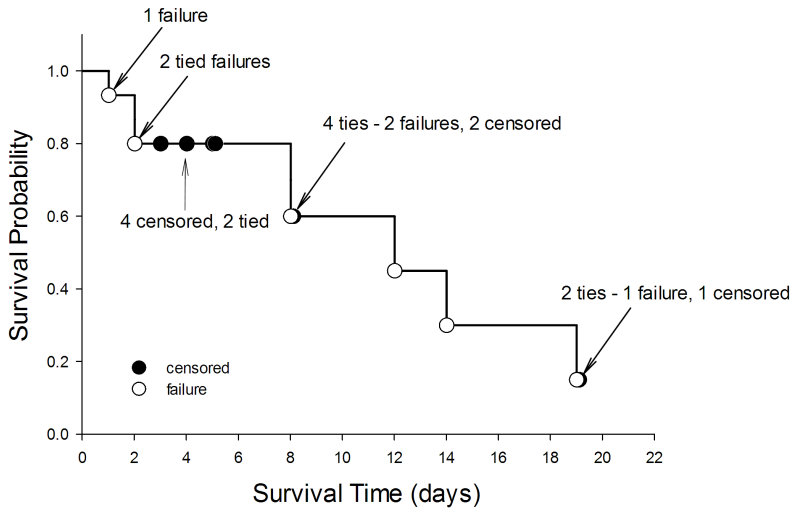


Figure 26: The Survival Graph

User-Defined Axis Scale

The USERAXIS.XFM transform is a specific example how to transform data to fit the user-defined axis scale.

This transform:

- Transforms the data using the new axis scale
- Creates Y interval data for the new scale

To use this transform to graph data along a `(log(log(100/Y))` Y axis, you can either use the provided sample data and graph, or begin a new notebook, enter your own data, and create your own graph using the data.

1. **To use the sample worksheet and graph,** double-click the graph page icon in the User Defined Axis Scale section of the Transforms Examples notebook. The User Defined Axis Scale worksheet appears with data in columns 1 through 3. The graph page appears with an empty graph with gridlines.
2. **To use your own data,** place your original X data in column 1, Y data in column 2, and the Y axis tick interval values in column 3. If your data has been placed in other columns, you can specify these columns after you open the USERAXIS.XFM file.
3. Press **F10** to open the **User-Defined Transform** dialog box, then click **Open** to open the USERAXIS.XFM transform. If necessary, change the `y_col`, `tick_col`, and `res` variables to the correct column numbers.
4. Click **Run**. The results are placed in columns 4 and 5, or the columns specified by the res variable.
5. If you opened the User Defined Axis Scale graph, view the page. The graph is already set up to plot the data and grid lines.
6. **To plot the transformed Y data,** plot column 1 as the X values versus column 4 as the Y values.
7. **To plot the Y axis tick marks,** on the **Property Browser** select:

   Page > **[Graph Title]** > **Axes** > **[Axis Title]** > **Ticks** > **Major Tick Labels**
8. On **Object Properties**, select Column 5 from the **Major Tick Intervals** drop-down list.

   To draw the tick labels, use the Y tick interval data as the tick label source.

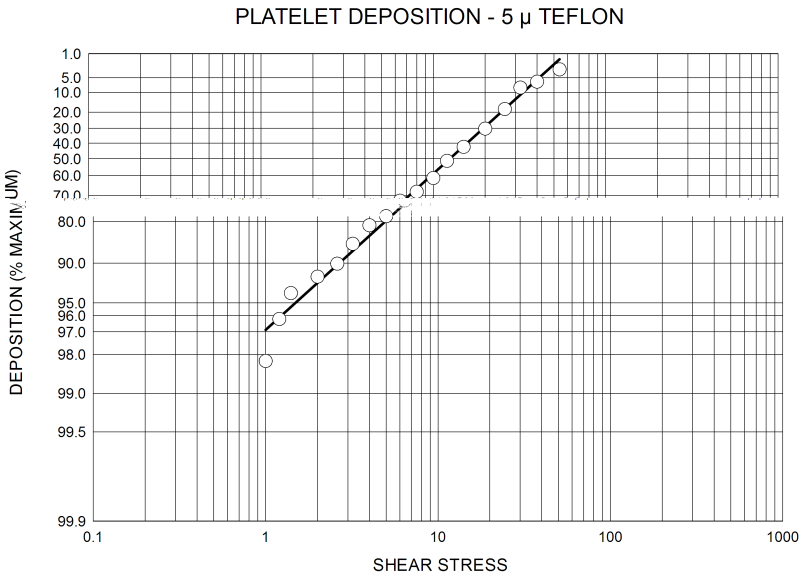


Figure 27: User-Defined Axis Scale Graph

Vector Plot

The VECTOR.XFM transform creates a field of vectors (lines with arrow heads) from data which specifies the X and Y position, length, and angle of each vector. The data is entered into four columns. Executing the transform produces six columns of three XY pairs, which describe the arrow body and the upper and lower components of the arrow head.

Other settings are:

- The length of the arrow head.
- The angle in degrees between the arrow head and the arrow body.
- The length of the vector (if you want to specify it as a constant).

To generate a vector plot, you can either use the provided sample data and graph or begin a new notebook, enter your own data, and create your own graph using the data.

1. **To use the sample worksheet and graph,** double-click the graph page icon in the Vector section of the Transform Examples notebook. The Vector worksheet appears with data in columns 1 through 4. The graph page appears with an empty graph.
2. **To use your own data,** enter the vector information into the worksheet. Data must be entered in four column format, with the XY position of the vector starting in the first column, the length of the vectors (which correspond to the axis units), and the angle of the vector, in degrees. The default starting column for this block is column one.
3. Press **F10** to open the **User-Defined Transform** dialog box, then click **Open** to open the VECTOR.XFM file in the XFMS directory.
4. If necessary, change the starting worksheet column for your vector data block xc.
5. If desired, change the default arrowhead length L (in axis units) and the Angle used by the arrowhead lines. This is the angle between the main line and each arrowhead line.
6. If you want to use vectors of constant length, set the l value to the desired length, then uncomment the remaining two lines under the Constant Vector Length heading.
7. Make sure that **Radians** are selected as the **Trigonometric Units** (they should be by default.
8. Click **Run** to run the transform. The transform produces six columns of three XY pairs, which describe the arrow body and the upper and lower components of the arrow head.
9. If you opened the Vector graph, view the page. The Line Plot with Multiple Straight Line appears plotting columns 5 through 10 as XY pairs.
10. **To plot the vector data using SigmaPlot,** create a Line Plot with Multiple Straight Line graph that plots columns 5 through 10 as three vector XY column pairs.

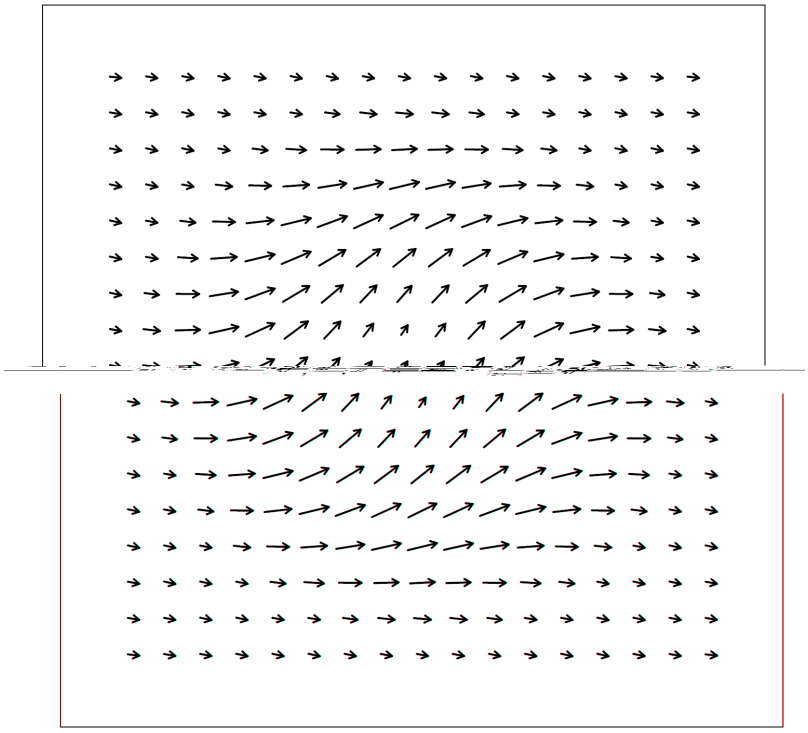


Figure 28: The Vector Graph

Z Plane Design Curves

The ZPLANE.XFM transform is a specific example of the use of transforms to generate data for a unit circle and curves of constant damping ratio and natural frequency.

The root locus technique analyzes performance of a digital controller in the z plane using the unit circle as the stability boundary and the curves of constant damping ratio and frequency for a second order system to evaluate controller performance.

Root locus data is loaded from an external source and plotted in Cartesian coordinates along with the design curves in order to determine performance.

Refer to *Digital Control of Dynamic Systems*, Gene. F. Franklin and J. David Powell, Addison-Wesley, pp. 32 and 104 for the equations and graph.

To calculate the data for the design curves, you can either use the provided sample data and graph, or begin a new notebook, enter your own data, and create your own graph using the data.

1. **To use the sample worksheet and graph,** double-click the graph page icon in the Z Plane section of the Transform Examples notebook. The Z Plane worksheet appears with data in columns 1 through 10. The Z Plane graph page appears with the design curve data plotted over some sample root locus data. This plot uses columns 1 and 2 as the first curve and columns 3 and 4 as the second curve.
2. **To use your own data,** place your root locus, zero, and pole data in columns 1 through 10. If your locus data has been placed in other columns, you can change the location of the results columns after you open the ZPLANE.XFM file.
3. **To plot the design curves of your data,** create a Line Plot with Multiple Spline Curves, then plot column 1 as the X data against column 2 as the Y data for the first curve and column 3 as the X data against column 4 as the Y data as the second curve.
4. Press **F10** to open the **User-Defined Transform** dialog box, then click **Open** to open the ZPLANE.XFM transform in the XFMS directory. If necessary, change the res variable to the correct column number.
5. Click **Run**. The results are placed in columns 11 through 20, or the columns specified by the res variable.
6. If you opened the Z Plane graph, view the page. The circle, frequency trajectory, and damping trajectory data is automatically plotted with the design data.
7. **To plot the circle data using SigmaPlot,** create Multiple Line Plots with Simple Spline Curves. For the first plot use column 11 as the X values versus column 12 as the Y values.
8. **To plot the frequency trajectory data (zeta),** plot column 13 versus column 14 and column 15 versus column 16 as the XY pairs.

9. **To plot the damping trajectory data (omega),** plot column 17 versus column 18 and column 19 versus column 20 as the XY pairs.

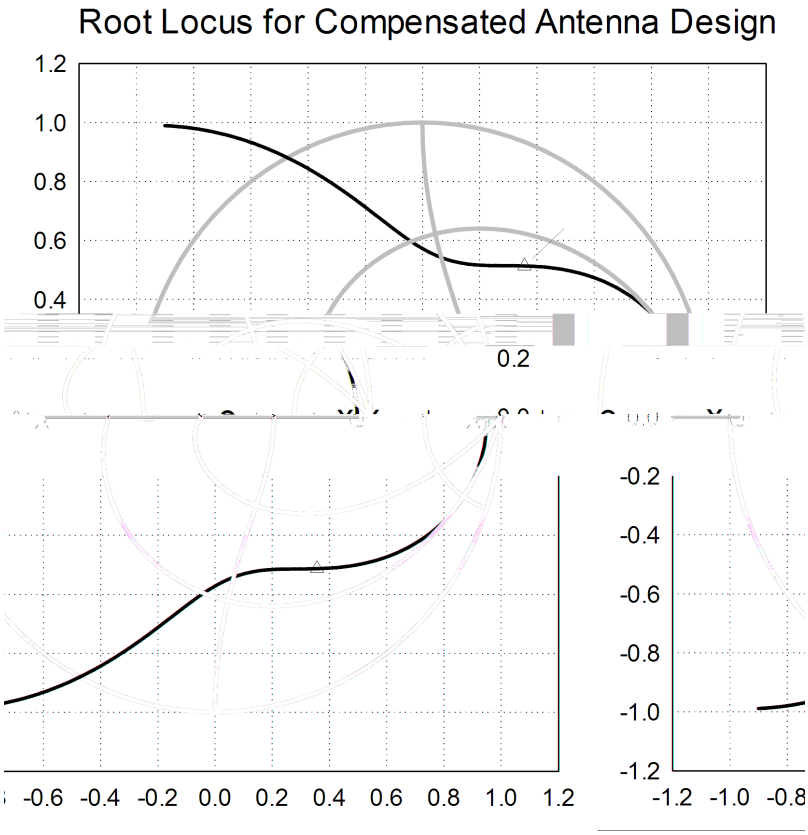


Figure 29: Z Plane Graph

Creating Histograms

Histograms are step, needle, or bar charts that represent counts of the data points that fall within specified ranges. The **Histogram Wizard** guides you through the steps in creating a histogram: generating frequency data, specifying the number of bins or intervals, and selecting a graph style.

The **Histogram Wizard** allows you to specify the number of bins into which to partition the source data. The range of each interval is identical; the total range is the data minimum to the data maximum. The number of bars, steps, or needles displayed is generally equal to the number of bins.

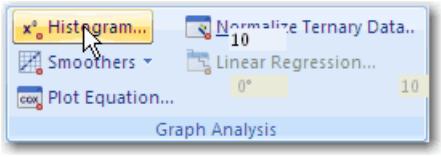
You can also create a histogram with an uneven bin width. For more information, see Creating Histograms on page 55.

Using the Histogram Wizard

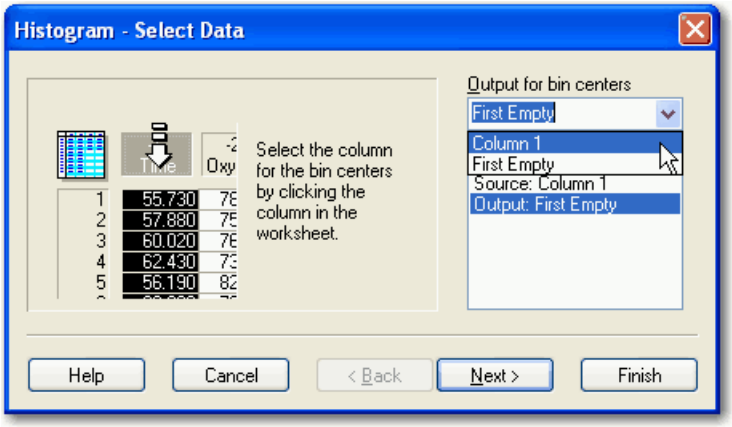
To use the Histogram Wizard:

1. Enter the data you want to analyze in an empty column of the active worksheet.

2. On the **Analysis** tab, in the **Graph Analysis** group, click **Histogram**.



3. Select the data for the histogram by choosing the appropriate column from the **Source data for histogram** drop-down list.



4. Select the column for the **Output for histogram** either from the drop-down list, or by clicking the column.
5. Select the column for the **Output for bin counts** either from the drop-down list, or by clicking the column.
6. Click **Next**. The **Histogram - Bin Options** panel appears, with **Automatic binning** already selected. The algorithm calculates the number of bins for representation, based upon the number of data points and the skewness of the data.

$$\text{Approximate Bins} = 1 + \left\lfloor \log_2(N) + \log_2\left(1 + \frac{|G_1|}{\sigma_{G_1}}\right) \right\rfloor$$

where N = number of non-missing points, G_1 is the adjusted Fisher-Pearson standardized moment coefficient, and $^{\sigma}G_1$ is the standard deviation of G_1.

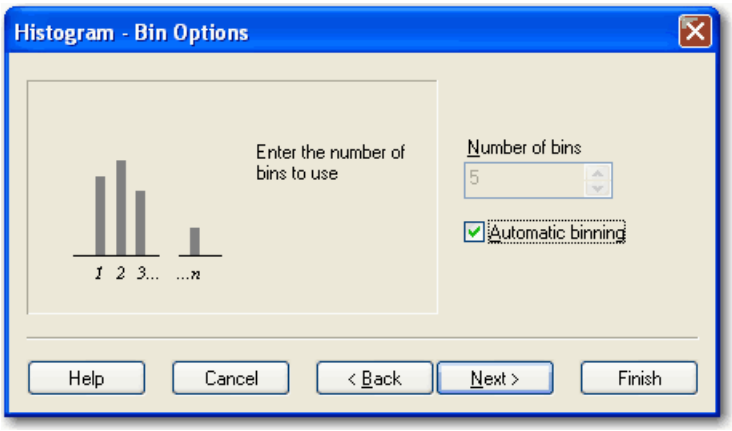


Figure 30: The Histogram Wizard – Bin Options Dialog Box

7. **To specify a different number of bins,** clear **Automatic binning** and select a number from the **Number of bins** list. You can enter values from 1 to 100.

8. **To scale the histogram values,** select one of three options from the **Normalization** drop-down list.

Option	Description
Bin	The bin height is the count or frequency of the data in the bin.
1	The bin height is the fraction of the total data count assigned to the bin.
100	The bin height is the percentage of the total data count assigned to the bin.

Note: NONE is the default value.

9. **To determine which edge of each bin is used for assigning data,** select one of two options from the **Select bin edge to include** drop-down list.

Option	Description
left	Any data that falls on the left edge of a bin is assigned to the bin.
right	Any data that falls on the right edge of a bin is assigned to the bin.

Note: The default is **left**.

10. Click **Next**.
11. Select a graph style from the **Graph Styles** list. A preview of the graph appears.

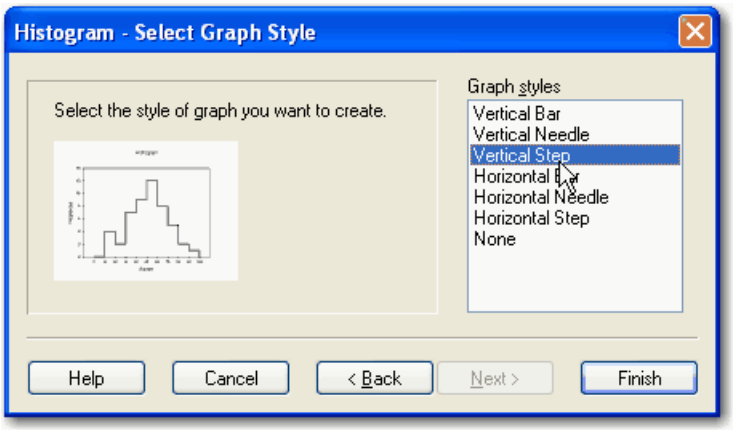


Figure 31: The Histogram Wizard – Graph Style Dialog Box

12. Click **Finish**.
    The graph appears on the active graph page, or a new page if the worksheet has no associated graph pages. The X axis representing the buckets is titled Raw Data. The Y axis representing the frequency or the number of data points in each bin, is titled Bin Count. Both use a linear scale.

    Note: If you choose **None**, SigmaPlot displays the worksheet with the output column containing the histogram frequency data.

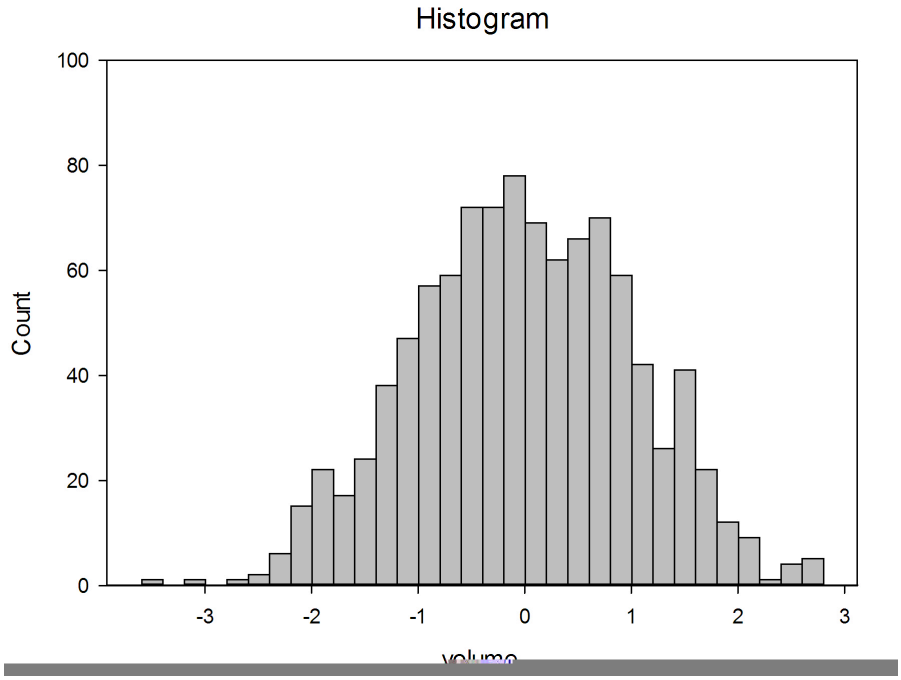


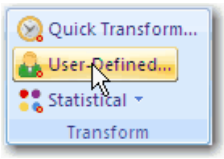
Figure 32: Example of a Histogram Created Using the Histogram Wizard

The Histogram Transform Function

If you need to use uneven bucket sizes for a histogram, use SigmaPlot's built-in *histogram transform function.*

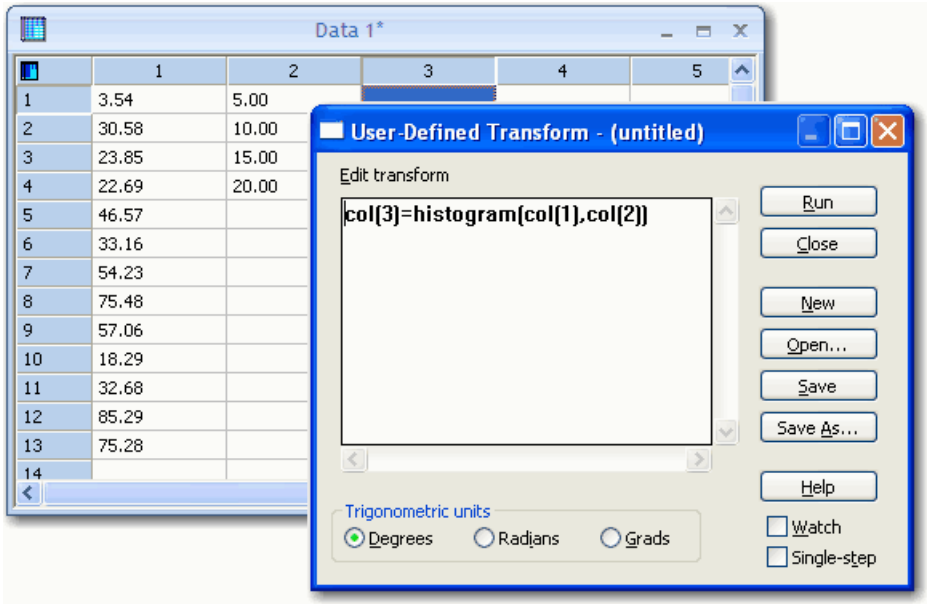
To use the histogram transform function:

1. Enter the data to analyze in column 1 the bin values in column 2 of the worksheet.
   Bin values are used as the upper bounds (inclusive) of the histogram interval ranges. The number of data points that fall within each specified range is counted. The number of histogram bars is equal to the number of interval upper bounds entered. The number of values that fall beyond the largest upper bound is also counted.
2. On the **Analysis** tab, in the **Transform** group, click **User-Defined**.



3. Enter the following transform into the **Edit Transform** box:

```
col(3)=histogram(col(1),col(2))
```



4. Click **Run**. The histogram data appears in column 3.
5. **To graph the data,** plot column 3 as a bar chart. For more information, see Creating 2D Plots.

Smoothing 2D and 3D Data

SigmaPlot smoothers are algorithms for smoothing sharp variations in dependent variable values within 2D and 3D data sets. You can also use smoothers to resample data to a rectangular grid of independent variable values.

You control the locations of the computed smoothed values. You can choose the raw data values of the independent variable(s) as the smoothing locations. You can also specify uniformly-spaced smoothing locations over the extent of the independent variable data.

Each smoothing method weights the data contained in a window surrounding the smoothing location. The radius of this window is called the bandwidth radius. A linear or nonlinear technique is then applied to the weighted data to compute each smoothed value.

The weight assigned to each data value in the window is determined by its normalized distance (u) from the smoothing location.

Choose one of the following smoothing methods:

- **Loess.** Applies the tricube weight function to weight the data. The smoother is polynomial of degree 1, 2, or 3. Use with 2D or 3D data.
- **Running Average.** Computes the average of the dependent values. Use with 2D or 3D data.
- **Running Median.** Computes the median of the dependent variable. Use with 2D or 3D data.
- **Negative Exponential.** Applies a Gaussian weight function to weight the data and a quadratic fit. Use with 2D or 3D data.
- **Bisquare.** Applies a bisquare weight function. Use with 2D or 3D data.
- **Inverse Square.** Applies a Cauchy weight function. Use with 2D or 3D data.
- **Inverse Distance.** Applies the weight function to the (x,y) data. Use with 3D data only.

You can find smoother method guidelines in the 2D and 3D Smoothers sections of **Samples.jnb**. For more information, see About SigmaPlot's User and Program Files.

Smoothing 2D Data

Use the Smooth 2D Data dialog box to remove undesired high-frequency data components, such as data contamination.

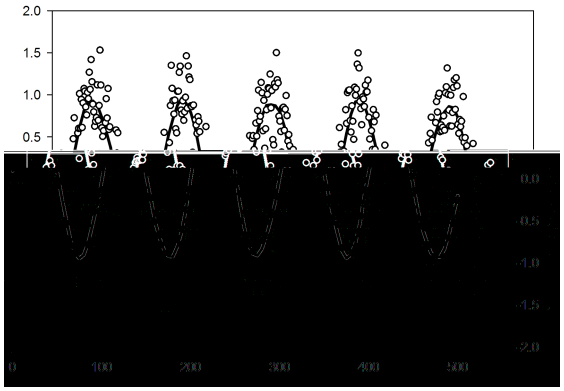
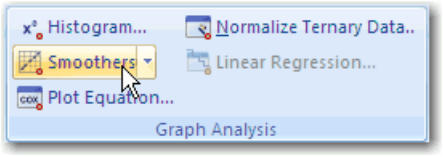
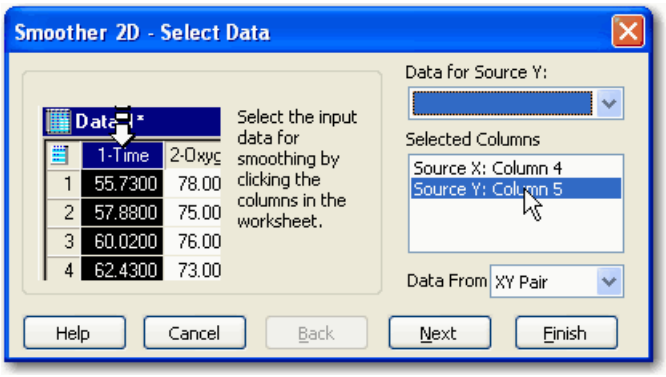


Figure 33: An example of noisy data and then its conversion. Note that the original noisy data points appear on the graph.

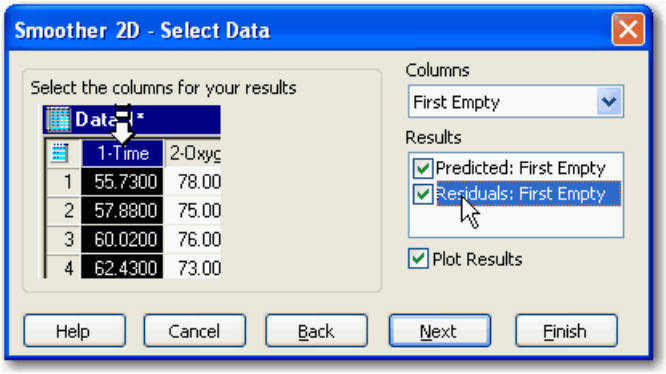
1. Select the worksheet columns by dragging the pointer over your data.
2. On the **Analysis** tab, in the **Graph Analysis** group, click **Smoothers**.



3. Click **Smooth 2D Data**
4. In the **Smoother 2D - Select Data** dialog box, click **Next**.



5. Select **Predicted: First Empty** from the **Results** list to compute a smoothed value for each data point.



6. Select **Residuals: First Empty** to differentiate between the smoothed value and the original Y value.
7. Accept **First Next Empty** as the standard default column in the **Columns** drop-down list.
8. Select **Plot Results** to create a grid of the computed smoothed values on the worksheet.
9. Select **Create Report** to create a record of the smoother settings used to create the results.
10. Click **Next**.
11. Accept **First Empty** as the default in the **Curve Data Column** list.

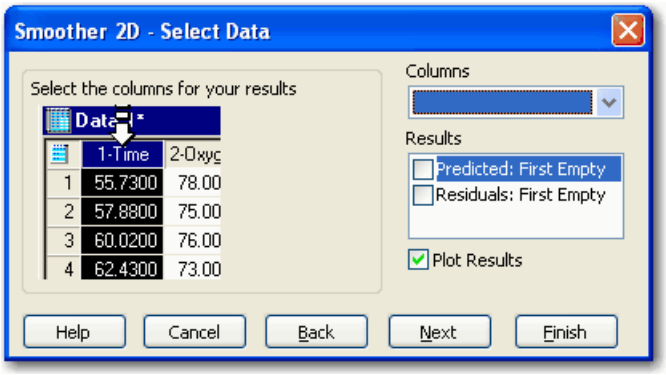


Figure 34: Selecting columns to display a grid of smoothed data on the worksheet.

12. Select **Create a new graph** to create a line plot using the grid of data which appears on the worksheet.
13. **To create another plot type and style,** clear **Create new graph**, and create the plot manually. For more information, see Creating 2D Plots.

14. Click **Finish**. The **Smooth 2D Data** dialog box appears.

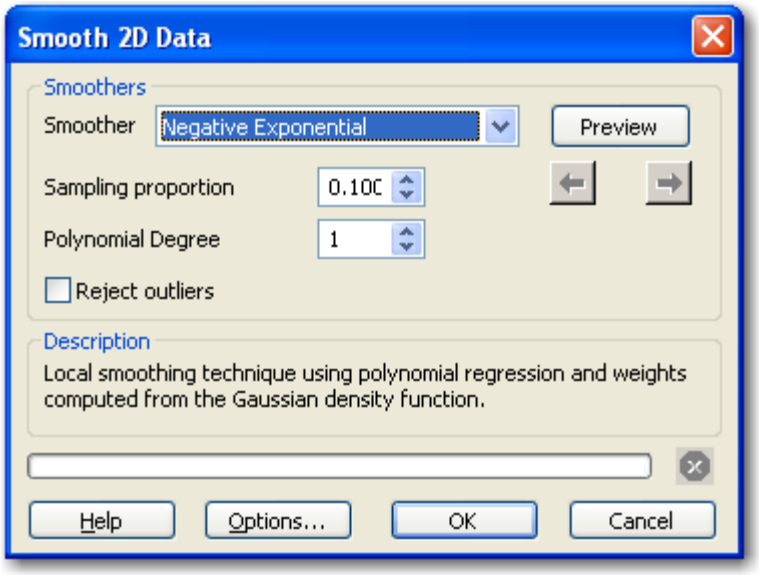


Figure 35: Selecting Smoothers from the Smooth 3D Data drop-down list

15. **To define smoothing parameters**, select a smoother type from the **Smoothers** drop-down list.
16. Set the **Sampling Proportion** to determine a fraction of the total number of data points used to compute each smoothed value.

    Note: The interpretation of the **Sampling Proportion** depends on the **Bandwidth Method**.

17. Set the polynomial degree from the **Polynomial Degree list**, if applicable.
18. Select **Reject Outliers** to reduce the effects of outlier points on the smoothed values.
19. To set smoothing options, click **Options**. The **Smoothed Curve Options** dialog box appears. For more information, see smoothing_2d_data.dita#setting_smoothed_curve_options.
20. **To preview and create the graph,** click **Preview** to see a preview of the graph.

    If the preview is not satisfactory, adjust the **Smoother** settings and options and click **Preview** again. Each time you preview, the settings are stored for subsequent review by clicking the right and left arrows.
21. Click **OK** to accept the preview.

    The graph appears with a line graph representing the smoothed data points. The original noisy data points also remain. The worksheet now contains the results of all selected computations.

    Tip: You can click the **Stop** button at the bottom of the **Smooth 2D Data** dialog box if you want to stop the process.

Setting Smoothed Curve Options

Use the **Smoothed Curve Options** dialog box to set the options for smoothing a 2D Curve.

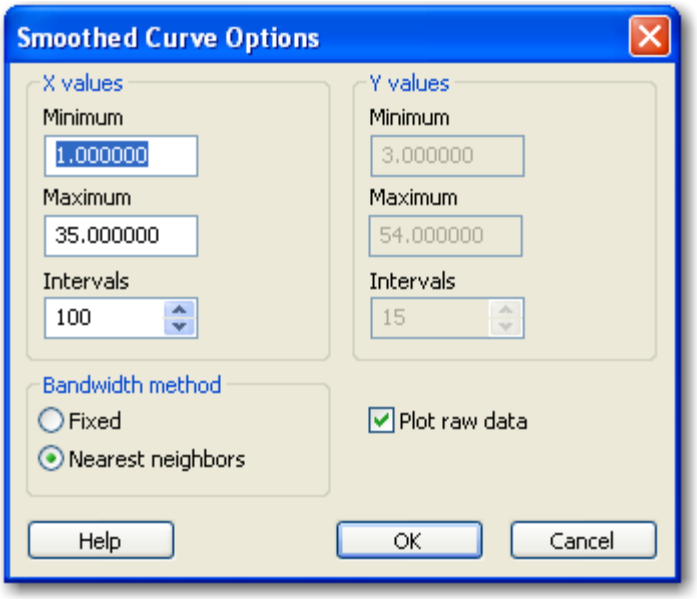


Figure 36: The Smooth Curve Options Dialog Box

1. Change the **Minimum** and **Maximum** for the X values to new beginning and ending values for the X ranges. For 2D smoothing, the Y values are the smoothed values, and therefore unavailable in the **Smoothed Curve Options** dialog box.
2. Set the **Bandwidth method** to either **Fixed** or **Nearest Neighbors**.
   - **Fixed.** Sets the same bandwidth radius the same at every smoothing location. The radius is computed by multiplying the **Sampling Proportion** value times half of the difference between the set **Minimum** and **Maximum** independent variables (X values). Select **Fixed** if the density of the observed data is relatively constant over the extent of its defined region.
   - **Nearest Neighbors.** Here the bandwidth radius depends on the smoothing location. The radius is equal to the maximum distance between the smoothing location and its nearest neighbors, as determined by the **Sampling Proportion** value. Select **Nearest Neighbors** for data that is clustered in some areas and sparse in others. For example, if there are 100 data points, enter .1 as the **Sampling Proportion** value to choose ten data points nearest the smoothing location.
3. Click **OK** to close the dialog box and return to the **Smooth 2D Data** dialog box.

Smoothing 3D Data

Use the **Smoother 3D** dialog box to smooth variations in 3D data. You can also re-sample 3D data to rectangular grid locations to create mesh plots and 3D contour plots from irregularly spaced data.

1. Select the worksheet columns by dragging the pointer over your data.

2. Click the **Analysis** tab, and then in the **Graph Analysis** group, select:
   Smoothers > **Smooth 3D Data**
   The **Smoother 3D - Select Data** dialog box appears.

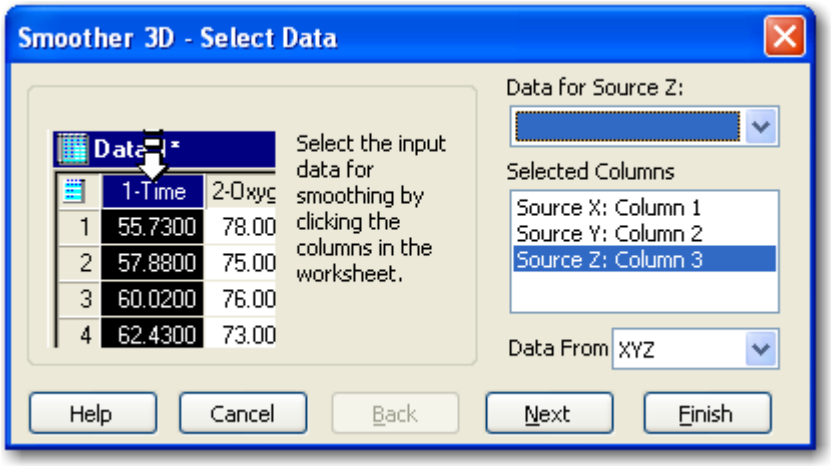


Figure 37: Selecting the Data Columns to Smooth from the Smoother 3D Dialog Box

3. Click **Next**.

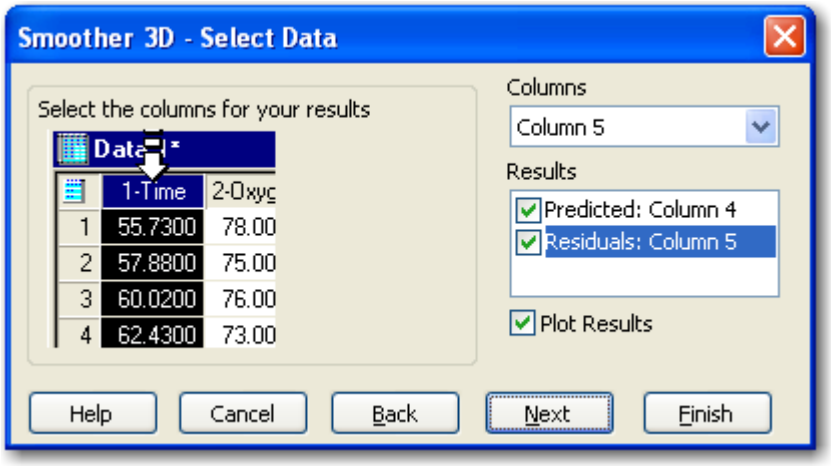


Figure 38: Selecting the Data Columns to Smooth from the Smoother 3D Dialog Box

4. **To select worksheet columns for your results,** select **Predicted: First Empty** from the **Results** list to compute a smoothed value at each data point.
5. Select **Residuals: First Empty** to differentiate between the smoothed value and the original Y value.
6. Accept **First Empty** as the standard default column in the **Columns** drop-down list.
7. Select **Plot Results** to create a grid of the computed data on the worksheet.
8. Select **Create Report** to create a record of the smoother settings used to create the results.
9. Click **Next**.

10. **To select columns to graph,** accept **First Empty** as the default in the **Columns** drop-down list.

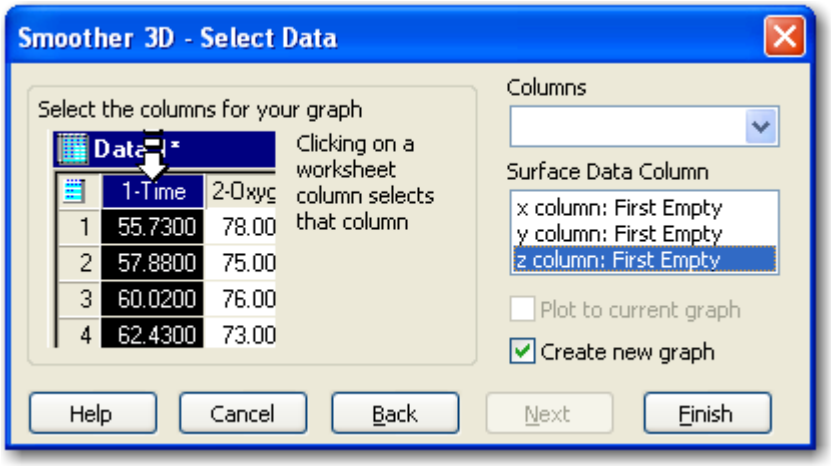


Figure 39: Selecting columns to display the grid of smoothed data

11. Select **Create a new graph** to create a mesh plot using the grid of data which appears on the worksheet. If you are creating a contour plot, clear **Create new graph**, and create the contour plot manually. Click **Finish**. For more information, see Creating Contour Plots.
12. In the **Smooth 3D Data** dialog box that appears, select a smoother type from the **Smoother** drop-down list.

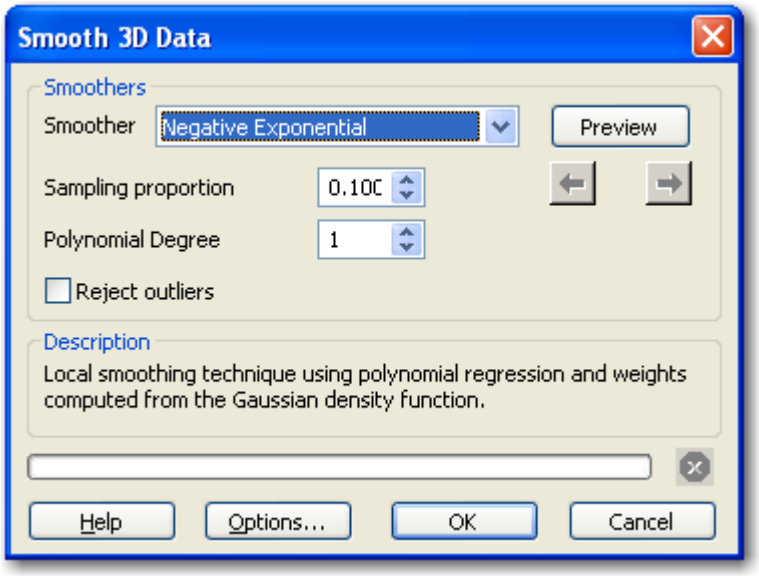


Figure 40: Selecting Smoothers from the Smooth 3D Data drop-down list

13. Set the **Sampling Proportion**, a fraction of a total number of data points used to compute each smoothed value.

18. Click **OK** to accept the preview.

    The graph appears, and the worksheet now contains the results of all selected computations.

    Tip: You can click the red **Stop** button at the bottom of the **Smooth 3D Data** dialog box to stop the process.

Setting Smoothed Surface Options

Use the **Smoothed Surface Options** dialog box to set the options for smoothing a 3D Curve.

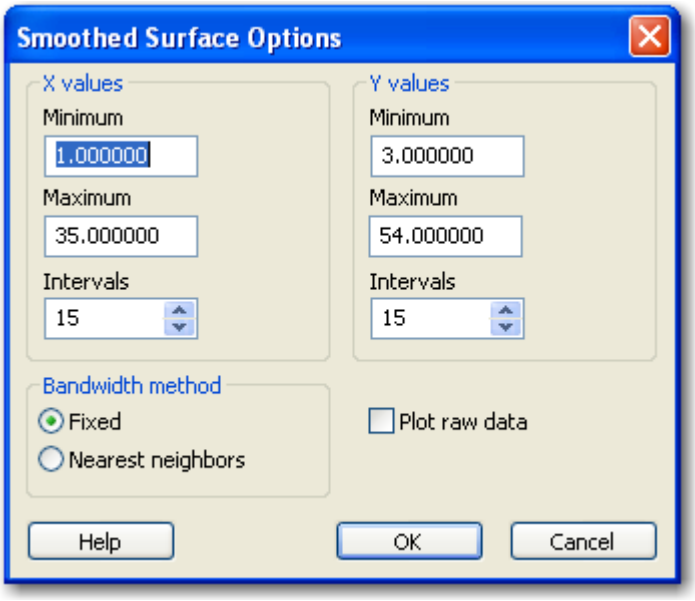


Figure 41: The Smooth Curve Options Dialog Box

1. Change the **Minimum** and **Maximum** for the X and Y values to new beginning and ending values for the X and Y ranges.
2. Set the bandwidth method to either Fixed or Nearest Neighbors.
   - **Fixed.** The bandwidth radius is the same at every smoothing location. The radius is computed by multiplying the **Sampling Proportion** value times half of the difference between the set **Minimum** and **Maximum** independent variables (X and Y values). Select **Fixed** if the density of the observed data is relatively constant over the extent of its defined region.
   - **Nearest Neighbors.** Here the bandwidth radius depends on the smoothing location. The radius is equal to the maximum distance between the smoothing location and its nearest neighbors, as determined by the **Sampling Proportion** value. Select **Nearest Neighbors** for data that is clustered in some areas and sparse in others.
3. Click **OK** to close the dialog box and return to the **Smooth 3D Data** dialog box.

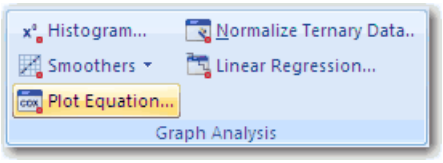
Plotting and Solving Equations

Use the Plot Equation dialog box to create and plot equations defined using the Transform language. You can use one of over 100 built-in equations, or create an equation of your own and save it to a notebook.

How to Plot and Solve an Equation

To create and plot an equation and save it to a notebook:

1. With the worksheet in view, on the **Analysis** tab, in the **Graph Analysis** group, click **Plot Equation**.

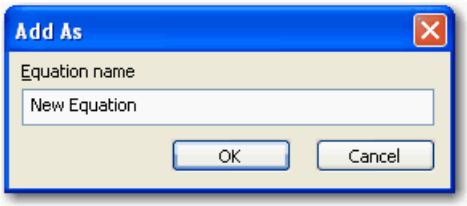


2. **To manually enter the equation,** from the **Name** drop-down list, select **Untitled**.
3. If necessary, delete the existing equation in the **f =** field, and then either type the equation, or click the **Functions Palette** button to open the **Functions Palette**. The **Functions Palette** provides immediate access to some of the most frequently used functions.

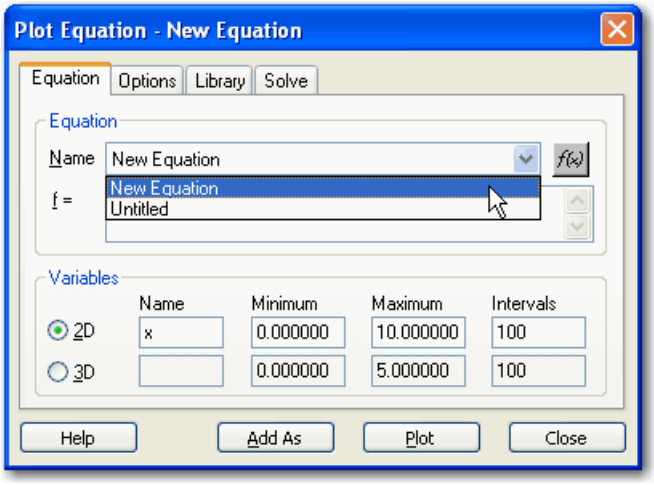
   You can also select one of the last ten used functions from the **Name** drop-down list. For more information, see Plotting Saved Equations on page 69.
4. From the **Variables** group box, select either **2D** or **3D**.
5. Set the independent variables using the **Name**, **Minimum**, **Maximum**, and **Intervals** boxes.

   - **Name.** Type the name of the independent variable(s).
   - **Minimum and Maximum.** Type the extent of the range of values for the corresponding independent variables.
   - **Intervals.** Set the number of intervals for sampling independent variables over a specified range.

   Tip: You can also select a column in the worksheet. The range of that column appears in the Minimum and Maximum edit boxes.
6. **To set the equation parameters,** click the **Options** tab. For more information, see Setting Equation Parameters on page 69.
7. Click **Add As**. The **Add As** dialog box appears.



8. Type the name of the equation in the **Equation Name** edit box.
9. Click **OK**. The equation name appears in the Name drop-down list on the **Equation** tab.



10. Click **Plot**. A graph page appears with the plotted equation, and the equation values appear in the worksheet.
11. Click **Close** to close the dialog box.

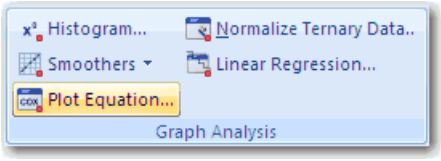
    If desired, you can add plot an equation and add it to the existing graph, or plot a new equation on a new graph page.

Plotting Equations onto Existing Graphs

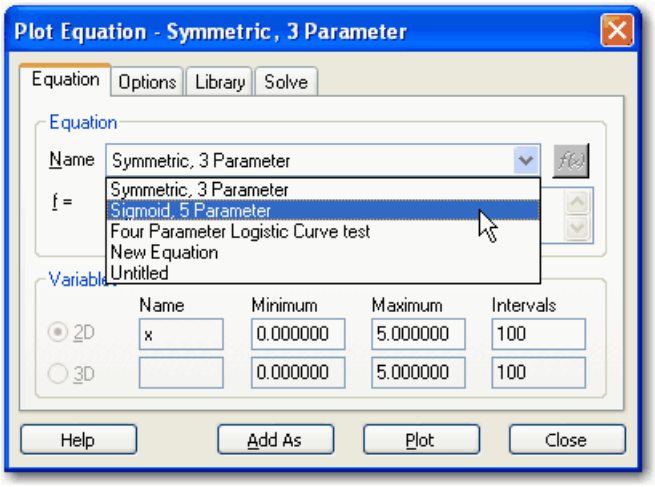
Use the **Plot Equation** dialog box to plot equations onto existing graphs. This is especially helpful if you want to see how the curves change by modifying the parameters.

To plot the equation:

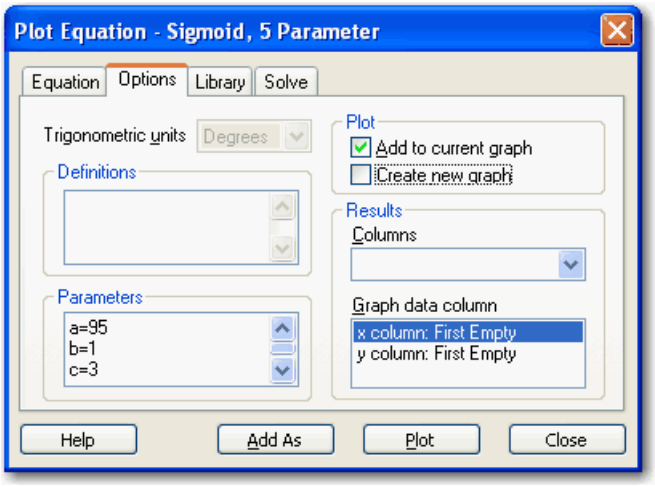
1. Select the graph.
2. On the **Analysis** tab, in the **Graph Analysis** group, click **Plot Equation**.



3. Click **Close** to close the **Plot Equations** dialog box.
4. In the **Plot Equation** dialog box, either manually enter the equation in the **f =** edit box, or choose an existing equation, or use the same equation as used previously if you want to change the parameters.



5. Click **Close** to close the **Plot Equations** dialog box.
6. **To set the equation parameters,** click the **Options** tab. For more information, see Setting Equation Parameters on page 69.

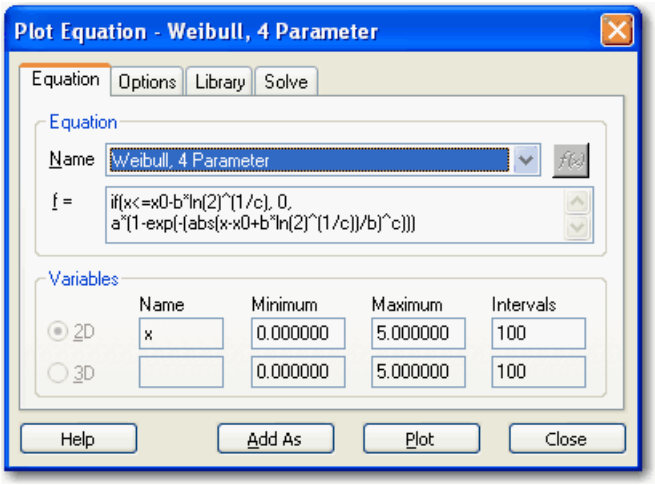


7. If you don't want to create a second graph page, select **Add to current graph** and clear **Create new graph**.
8. Click **Close** to close the **Plot Equations** dialog box.

Setting Equation Parameters

All equations that you create or use from the Standard. sm

3. Select an equation category from the **Equation category** drop-down list. The items that appear in the **Equation category** drop-down list are sections in the Standard.jfl library. Below, in the Equation Name list, are items that appear under that section name in the notebook.
4. Select an equation from the **Equation name** list.
5. Click **Select**. The **Equation** tab appears with the selected equation displayed in the **Name** drop-down list.



Some of the settings for SigmaPlot's built-in equations in the Standard.jfl library are read-only. To modify a built-in equation, click **Add A**s to create an equation based on the built-in equation.

6. Click **Plot**. A graph page appears with the plotted equation, and the equation values appear in the worksheet.
7. Click **Close** to close the **Plot Equation** dialog box.

Solving Equations

Use the **Equation Solver** on the **Plot Equations** dialog box to evaluate mathematical expressions for functions and to solve equations.

The **Equation Solver** uses the expression entered in the **Equation** tab on the **Plot Equations** dialog box as the basis for its results. This expression then appears on the **Solve** tab for evaluation.

To solve an equation:

1. On the **Analysis** tab, in the **Graph Analysis** group, click **Plot Equation**

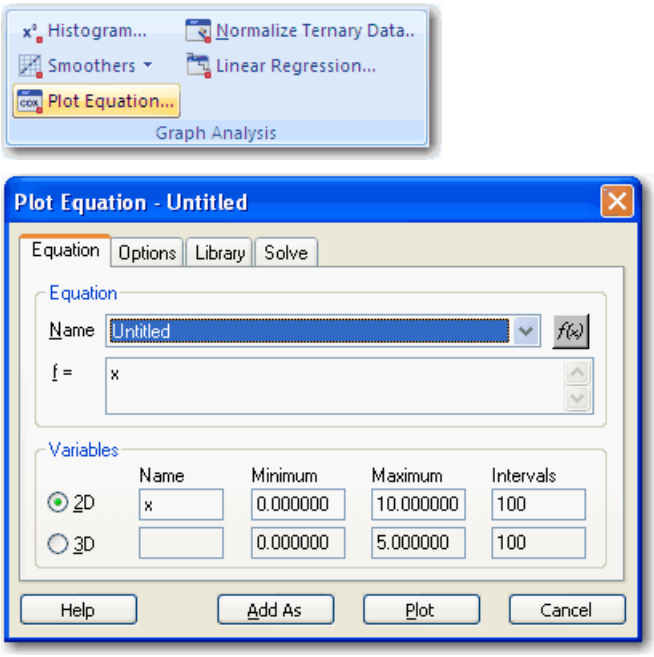
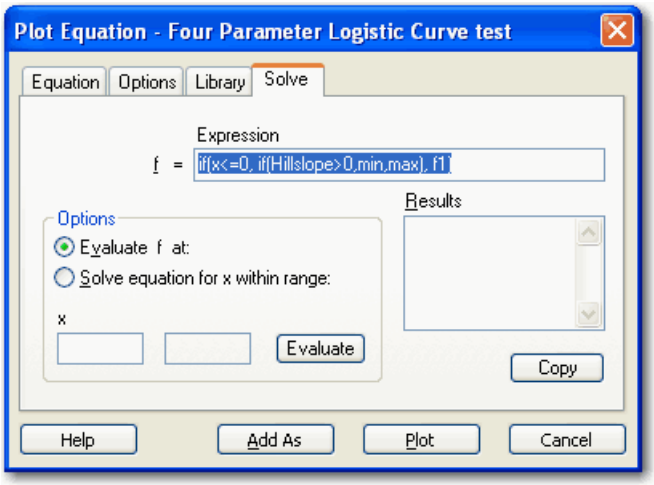


Figure 42: Plot Equation Dialog Box

2. Click the **Equation** tab, and enter an equation in the **f =** box. You can also select one of the last ten used functions from the **Name** drop-down list, or you can choose any of the built-in parameterized equations used by the **Regression Wizard**. Select these equations from the **Library**, too. For more information, see Plotting Saved Equations on page 69.
3. Click the **Solve** tab. The entered equation appears in the **f =** box on the **Solve** tab.



4. Under **Options**, select the mode of operation. You can select from one of the following:

   - **Evaluate F at.** Enter a numerical value for each variable that occurs in the expression in the boxes that appear at the bottom of the dialog box.
   - **Solve equation for x within range.** Enter a numerical value into the box which appears to the left of the expression (the default value is 0) to complete the definition of the equation. You must also enter limits for a range of values of the equation variable. The default range limits are taken from the values entered on the **Equation** tab.

   The Solver is only available for expressions containing a single independent variable, although any number of parameters can be present.

Results Box Tips and Tricks

- The **Results** box keeps a tally of all evaluation and solving results relative to the given expression. If you alter this expression on the **Equation** tab or select a new plot expression, the **Results** box appears with no text. Modifying the expression also clears the other boxes on the **Solve** tab.
- Click **Copy** to place the entire contents of the **Results** box onto the Clipboard.
- You can annotate the results in the **Results** box. All annotations are preserved when your perform further computations using the same expression.
- In addition to displaying the results of evaluating functions and solving equations, the **Results** box also displays estimates for any singularities found in the course of solving an equation. Singularities are values of the expression variable (in the given range) where the expression is undefined. When you perform a computation, a label precedes the values in the Results box to indicate the type of output displayed.

Equation Solving Guidelines

Sometimes the solutions to an equation `0 = f(x)` are not obvious and the basic methods for solving it are unavailable. If this is the case, then the simplest way to estimate the location of solutions is to:

1. Using the **Plot Equations** dialog box, graph the function equation `y = f(x)`.
2. Observe where the graph intersects the x-axis.

   This technique aids in determining range limits for the independent variable in the **Function Solver** (Solve tab of the Plot Equation dialog box).

   If the distance between two solutions of an equation is small relative to the size of the range, then the **Function Solver** may not return both solutions. The resolution of the solutions is approximately two orders of magnitude less than the size of the range. You can obtain higher resolution by adjusting the range limits to reduce the range size.

   There is particular difficulty, due to roundoff error, in determining solutions to `0 = f(x)` at points where the graph of `y = f(x)` does not cross the x-axis, but lies on one side of it.

   An example of this situation is the graph of `y = x^3+x^2 at x = 0`. Although in many cases, as with the above equation, the Function Solver provides the solution, in some cases, however, the solution will not be found and recorded in the Results box.

   If you suspect that there is such a solution and the Function Solver does not find it, then try the following technique for approximating the solution:
3. Alter the value for the left side of the equation by a small amount.
4. Resolve the equation.

   This is equivalent to slightly shifting the graph of the equation up or down until it lies on both sides of the axis. In general, the **Results** edit box then reports two solutions that are very close together. As smaller amounts are used to adjust the left side of the equation, these two solutions are seen to converge to one solution.

   As an example, try solving the equation `0 = sin(2*x)*cos(3*x)` over the range from `x = 1` to `x = 2`. The **Function Solver** will indicate that there are no solutions. Using the above technique will yield solutions that are close to the true solution of PI/2.

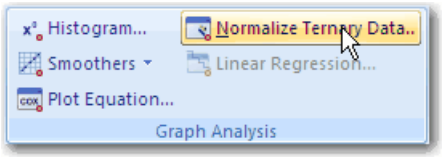
   Spurious Solutions. A less frequent problem involves the appearance of spurious solutions. Due to the limits of floating point numbers, the value of an expression `f(x)` at `x = a` might compute to zero even if `x = a` is not a true solution to `0 = f(x)`. This situation commonly arises when the graph of `y = f(x)` is very "flat" near a point where it intersects the x-axis.
   For example, consider the equation `0 = x^201`. If you solve this equation over the range from `x=0` to `x=1`, then the **Function Solver** will return 13 solutions even though the only true solution is `x = 0`. This is because each of 13 results raised to the 201st power is equal to zero in the machine's floating point representation.

Normalizing Ternary Data

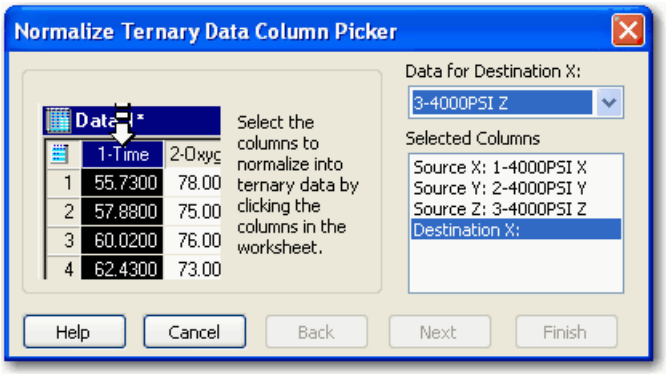
To create a ternary graph using data whose sum is not 100% or 1, you need to convert the raw XYZ data into normalized ternary triplet data by using the Normalize Ternary Data transform.

How to Normalize Ternary Data

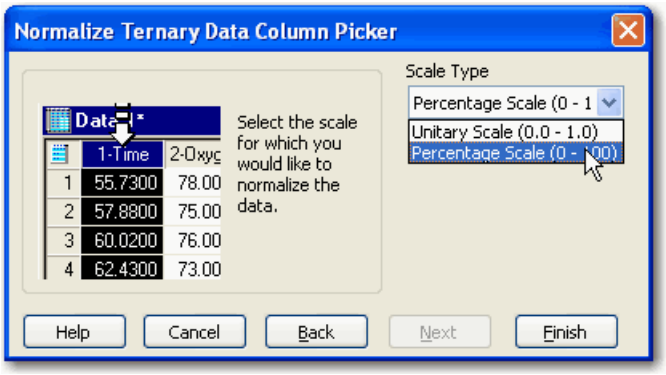
1. On the **Analysis** tab, in the **Graph Analysis** group, click **Normalize Ternary Data**.



2. In the **Normalize Ternary Data Column Picker** dialog box, select the column with the original X data from the worksheet or the **Data Source** list. The selected column is assigned as the X Source in the **Selected Columns** list.



3. Select the Y data source.
4. Select the columns from the worksheet data.
5. Select the X, Y, and Z data destination columns in the worksheet.
6. Click **Next**.
7. Select the type of scale from the **Scale Type** drop-down list.



8. Click **Finish**.

Plotting and Modifying Regression Lines

You can automatically compute and draw linear and polynomial regressions with confidence and prediction intervals. The regression equation can be computed using all the data in a plot, or individually for each curve in a multiple-curve plot. Polynomial curves can be fitted up to the 10th order.

Regressions for column averaged data are computed using all the data from the columns, not just from the mean value. Regressions are computed and drawn linearly on nonlinear (*e.g.,* log, probability, *etc.*) axis scales.

Regression equation coefficients, R^2 values, and predicted values can be viewed and copied to the Clipboard.

To perform nonlinear regressions and curve fits, such as sigmoidal, exponential, and peak functions, use SigmaPlot's Regression Wizard. The Regression Wizard provides an extensive set of equations for curve fitting.

Modifying and Adding Linear Regression Lines

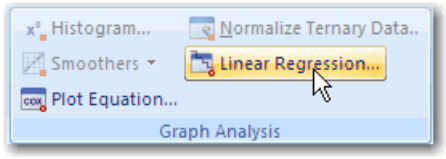
Add a first order regression to a graph by selecting one of the graph styles that has a regression. These styles include:

- Simple Regression
- Multiple Regression
- Simple Error Bars and Regression
- Multiple Error Bars and Regression

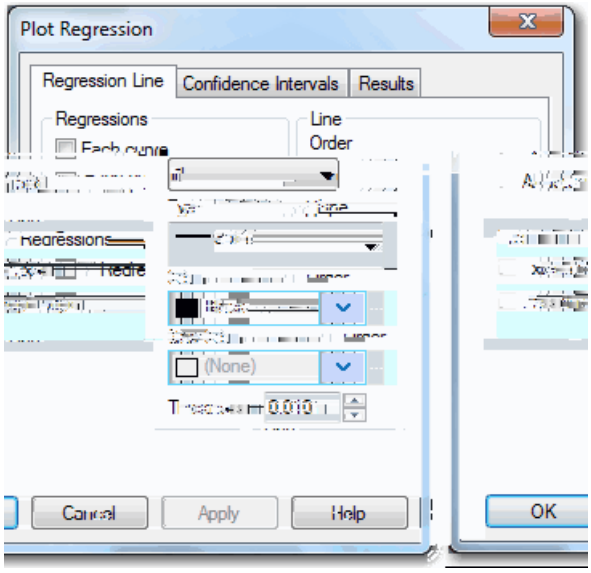
For more information, see About the Regression Wizard on page 80.

To modify or add a regression to a plot:

1. Click the plot to select it.
2. On the **Analysis** tab, in the **Graph Analysis** group, click **Plot Regression**.



3. In the **Plot Regression** dialog box, click the **Regression Line** tab.



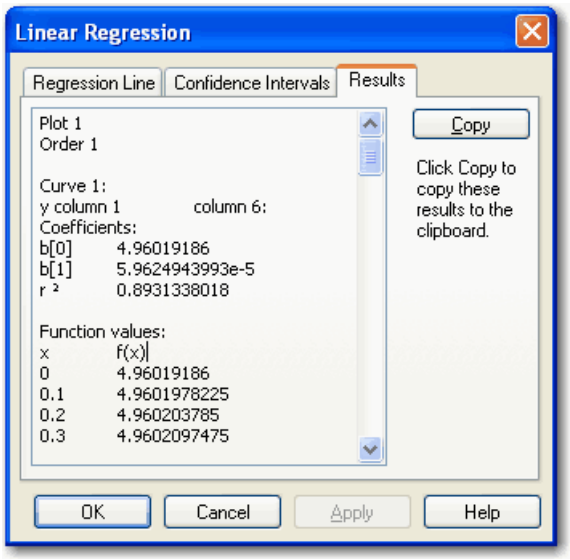
4. Under **Regressions**, select either **Each Curve** to draw a regression for the data in each curve of the selected plot, or **All data in plot** to draw a single regression for all of the data in the selected plot from the **Regressions** group box.

   If neither box is selected a regression is not drawn. If both boxes are selected, regressions are drawn for each curve and for all the data in the plot.
5. Under **Line**, select the desired regression order from the **Order** drop-down list.
6. Click **OK**.

Viewing and Saving Regression Equation Results

If you want to view and save the coefficients of the regression(s), select the **Results** tab of the **Plot Regression** dialog box. The **Results** tab appears displaying regression equation results.

The regression equation coefficients, correlation coefficient R^2, and function results are displayed for each regression curve computed. If you computed confidence and prediction intervals, these values are also displayed



Click **Copy** to copy the results and paste them into the worksheet, a report, or any other Windows application. For more information, see Linear Regression, Confidence, and Prediction Calculation on page 76.

Adding Confidence and Prediction Intervals

SigmaPlot can draw lines which describe either the 95% or 99% confidence and prediction intervals around a regression line.

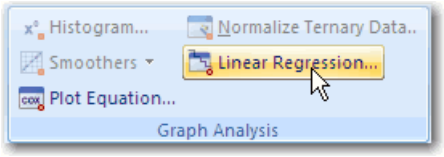
Confidence intervals (or confidence line), also called the confidence interval for a regression, describe the range where the regression line values will fall a percentage of the time for repeated measurements.

Prediction intervals, also called the confidence interval for the population, describe the range where the data values will fall a percentage of the time for repeated measurements.

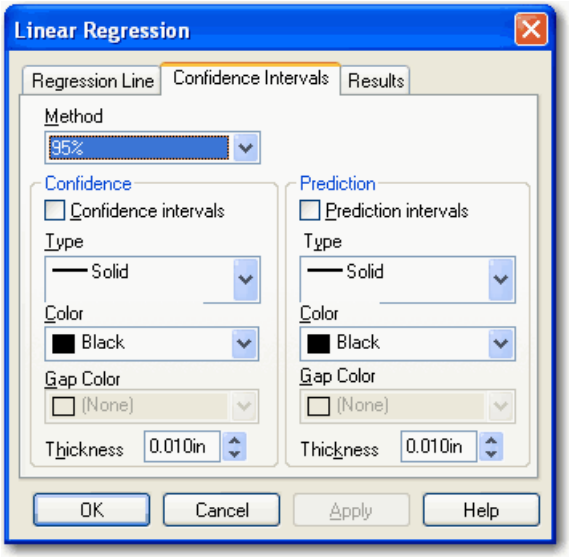
Remember: You must compute a regression in order to compute confidence and prediction lines.

To add prediction and confidence lines:

1. On the **Analysis** tab, in the **Graph Analysis** group, click **Linear Regression**.



2. Click the **Confidence Intervals** tab.



3. Choose the method of prediction to use from the **Method** drop-down list. Select either **95%** or **99%** for confidence and prediction intervals.
4. Click **OK**.

Linear Regression, Confidence, and Prediction Calculation

Regression Calculation

SigmaPlot linear regression uses the least squares method to construct a fit a set of data points (x_i, y_i) i = 1, ..., n by a polynomial of degree p where: $y=\beta_0+\beta^x{}_1+\beta_2x^2+\beta_px^p$

In vector-matrix notation this problem is formulated as $Y=X\beta+\varepsilon$:

where the n * 1 vector containing the y_n data is: $Y=[\,y_1 \quad y_2 \quad \quad y_n\,]$

$$X=\begin{matrix}1 & x_1 & x^2{}_1 & & x^p{}_1\\ 1 & x_2 & x^2{}_2 & & x^p{}_2\\ 1 & x_n & x^2{}_n & & x^p{}_n\end{matrix}$$

and the n * (p +1) design matrix is:

β is a (p + 1) * 1 vector of parameters to be estimated:

$\beta=[\,\beta_0 \quad \beta_1 \quad \quad \beta_p\,]$

ϵ is an n x 1 vector of residuals.

The solution for the least squares estimates of the parameters
β
is: $b=(XX)_{-1}XY$

where X' denotes the transpose of X.

SigmaPlot uses the Cholesky decomposition to invert the $X'Y$matrix. (see Dongarra, J.J., Bunch, J.R., Moler, C.B., and Stewart, G.W., *Linpack User's Guide*, SIAM, Philadelphia, 1979). This produces the regression curve: $y=b_0+b_1x+b_2x^2+b_px^p$

For further details on matrix linear regression, refer to chapter 2 of Draper, Norman, and Smith, Harry, *Applied Regression Analysis*, Second Edition, John Wiley & Sons, Inc., New York, 1981.

Confidence Interval Calculation

Given a set of n data points (x_i, y_i) from two columns in the worksheet, SigmaPlot computes the p^{th} order polynomial regression: $y=b_0+b_1x^0+b_2x^2+b_px^p0$where ($b_0$, b_1, ..., b_p) are the p + 1 estimated parameters and $\hat{y}_0$ is the y value predicted for any x_0.

The confidence interval for this calculated regression is defined by the two confidence limits: $y_0 \pm t(n-p-1)s\sqrt{X_0(XX)^{-1}X_0}$ where X_0 is the (p +1) * 1 vector defined by $X_0 = [1 \quad x_0 \quad x2_0 \quad xp_0]$

X is the n * (p +1) design matrix:

$$X = \begin{bmatrix} 1 & x_1 & x^2_1 & x^p_1 \\ 1 & x_2 & x^2_2 & x^p_2 \\ 1 & x_n & x^2_n & x^p_n \end{bmatrix}$$

s is obtained from the variance about the regression $s^2 = \sum_i \frac{(y_i - y_0)^2}{n-2}$

and the t value for n - p - 1 degrees of freedom and the standard normal percentile equivalent z (z = 1.96 or 2.576 for 95% and 99% confidence intervals respectively) is computed from a six term rational polynomial approximation taken from Sahai, H. and Thompson, W., “Comparisons of Approximation to the Percentile of t,χ^2 , and F Distributions,” *Journal of Statistical Computation and Simulation*, 1974, Vol. 3, pp. 81-93.

Prediction Interval Calculation

The prediction interval is calculated using the following equation: $y_0 \pm t(n-p-1)s\sqrt{1+X_0(XX)^1X_0}$

Chapter 3

Nonlinear Regression

Topics:

Regression is most often used by scientists and engineers to visualize and plot the curve that best describes the shape and behavior of their data.

Regression procedures find an association between independent and dependent variables that, when graphed on a Cartesian coordinate system, produces a straight line, plane or curve. This is also commonly known as curve fitting.

The *independent* variables are the known, or predictor, variables. These are most often your X-axis values. When the independent variables are varied, they result in corresponding values for the *dependent*, or response, variables, most often assigned to the Y-axis.

Regression finds the equation that most closely describes, or fits, the actual data, using the values of one or more independent variables to predict the value of a dependent variable. The resulting equation can then be plotted over the original data to produce a curve that fits the data.

About the Regression Wizard

SigmaPlot uses the Regression Wizard to perform regression and curve fitting. The Regression Wizard provides a step-by step guide through the procedures that let you fit a known function to your data and then automatically plot the best-fit curve and produce statistical results.

The Regression Wizard simplifies curve fitting. There is no need to be familiar with programming or higher mathematics. The large library of built-in equations are graphically presented and organized by different categories, making selection of your models straightforward. Built-in shortcuts let you bypass all but the simplest procedures; fitting a curve to your data can be as simple as picking the equation to use, then clicking a button.

Tip: For more complicated curve fitting, try using the Dynamic Fit Wizard. Dynamic Curve Fitting on page 115

Use the Regression Wizard to:

- **Select the function describing the shape of your data.** SigmaPlot provides over 100 built-in equations. You can also create your own custom regression equations. Regression Equation Libraries and Notebooks on page 107
- **Select the variables to fit to the function.** You can select your variables from either a graph or a worksheet.
- **Evaluate and save your results.** You can automatically plot the resulting curves on a graph, and save statistical results to the worksheet and text reports.

The Regression Wizard is also compatible with older .FIT files. Opening .FIT Files on page 81

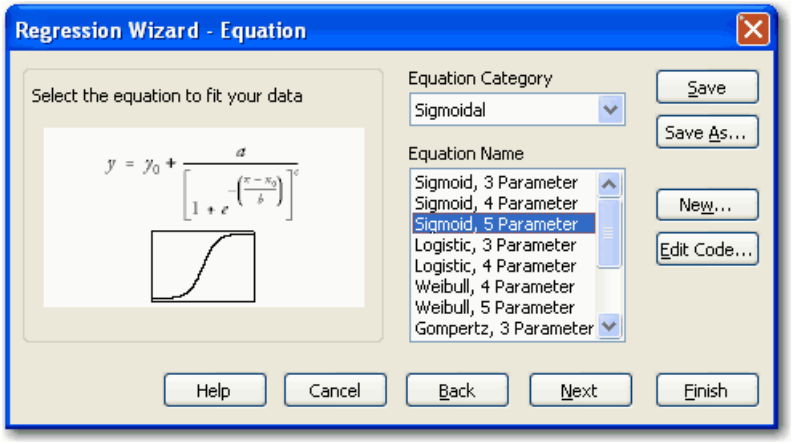


Figure 43: Selecting an Equation from the Regression Wizard

About the Curve Fitter

The *curve fitter* works by varying the parameters (coefficients) of an equation, and finds the parameters which cause the equation to most closely fit your data. Both the equation and initial parameter values must be provided. All built-in equations have the curve equation and initial parameters predefined.

The curve fitter accepts up to 500 equation parameters and ten independent equation variables. You can also specify up to 50 parameter constraints, which limit the search area of the curve fitter when checking for parameter values.

The regression curve fitter can also use weighted least squares for greater accuracy.

Curve-fitting Algorithm

The SigmaPlot curve fitter uses the Marquardt-Levenberg algorithm to find the coefficients (parameters) of the independent variable(s) that give the *best fit* between the equation and the data.

This algorithm seeks the values of the parameters that minimize the sum of the squared differences between the values of the observed and predicted values of the dependent variable

$$\sum_{i=1}^{n} w_i (y_i - \hat{y}_i)^2$$

Where y_i is the observed, $\hat{y}_i$ is the predicted value, w_i is the weight of the dependent variable..

This process is *iterative*—the curve fitter begins with a guess at the parameters, checks to see how well the equation fits, then continues to make better guesses until the differences between the residual sum of squares no longer decreases significantly. This condition is known as *convergence*. For more information, see References for the Marquardt-Levenberg Algorithm on page 81.

References for the Marquardt-Levenberg Algorithm

Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T. (1986). *Numerical Recipes.* Cambridge: Cambridge University Press.

Marquardt, D.W. (1963). An Algorithm for Least Squares Estimation of Parameters. *Journal of the Society of Industrial and Applied Mathematics,* 11, 431-441.

Nash, J.C. (1979). *Compact Numerical Methods for Computers: Linear Algebra and Function Minimization.* New York: John Wiley & Sons, Inc.

Shrager, R.I. (1970). Regression with Linear Constraints: An Extension of the Magnified Diagonal Method. *Journal of the Association for Computing Machinery,* 17, 446-452.

Shrager, R.I. (1972). Quadratic Programming for N. *Communications of the ACM*, 15, 41-45.

Opening .FIT Files

To open old curve fit (.fit) files:

Click the **Main Button** and then click **New**.

Select **SigmaPlot Curve Fit** as the file type. .fit files are opened as a single equation in a notebook.

You can also open .fit files from the **Library** panel of both the **Regression Wizard** and the Dynamic Fit Wizard. Dynamic Curve Fitting on page 115

Adding .FIT Files to a Library or Notebook

Add these equations to other notebooks by copying and pasting. To add them to your regression library, open the library notebook (Standard.jfl), then copy the equation and paste it into the desired section of the library notebook.

You can also create your own library by simply combining all your old .fit files into a single notebook, then setting this notebook to be your default equation library.

Tip: Sections appear as categories in the library, so create a new section to create a new equation category.

.FIT files as well as new equations do not have graphic previews of the equation.

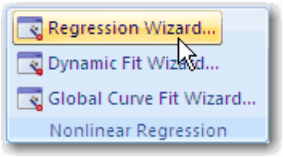
Using the Regression Wizard

The first step for using the Regression Wizard is to select the data source.

1. View the page or worksheet with the data you want to fit.
2. **If you select a graph,** right-click the curve you want fitted, and on the shortcut menu, click **Fit Curve**.

Remember: If you are running a regression from the graph page, make sure you select the plot itself, not the graph, or **Fit Curve** will not appear on the shortcut menu.

3. **If you are using a worksheet**, select the variables in the worksheet you want to fit, then on the **Analysis** tab, in the **Nonlinear Regression** group, click **Nonlinear Regression**.



Selecting the Equation to Use

Select an equation from the **Equation Category** and **Equation Name** drop-down lists. You can view different equations by selecting different categories and names. The equation's mathematical expression and shape appear to the left.

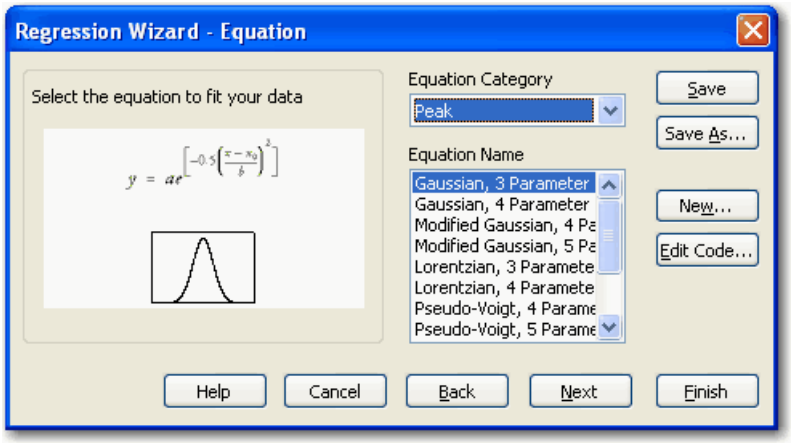


Figure 44: Selecting an Equation Category and Equation Name

If the equation you want to use isn't on this list, you can create a new equation. You can also sbrowse other notebooks and regression equation libraries for other equation.

Tip: SigmaPlot remembers the equation for the next time you open the Regression Wizard.

If the **Finish** button is available, you can click it to complete your regression. If it is not available, or if you want to further specify your results, click **Next**.

Selecting the Variables to Fit

1. Click **Next** to open the **Variables** panel. From here, you can select or re-select your variables.
2. If you pick variables from a worksheet column, you can also set the data format. Variable Options on page 90 When you have selected your variables, you can either click **Finish**, or click **Next** to view the Initial Results.

Viewing Fit Results

The fit results also appear if you receive a warning or error message about your fit. If you wish to modify the remainder of the results that are automatically saved, click **Next**. Otherwise, click **Finish**. The subsequent panels provide options for the output data.

Setting Numeric Output Options

Use the **Regression Wizard - Numeric Output Options** panel to:

- Decide which results are saved to the worksheet.
- Generate a text report of the regression.
- Save a copy of the regression equation to the notebook.

1. Select which results you want to keep from the **Results** list. These settings are remembered between regression sessions.
2. To set the options for the report, click **Report**.
3. Click **Next** to set the graph options.
4. Click **Finish**. A nonlinear regression report appears.

Setting Graph Options

This panel is only available if your fit equation has at most two independent variables.

1. If you selected your variables from a graph, select **Add curve to** to automatically add the equation curve to that graph.

   You can also plot the equation on any other graph on that page by selecting one from the drop-down list.
2. Select **Create new graph** to create a new graph of the original data and fitted curve.

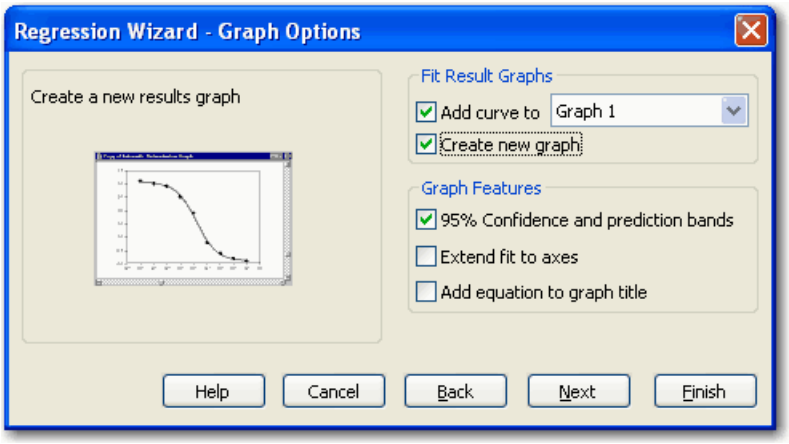


Figure 45: Selecting the results to graph. These settings are retained between sessions.

3. Select **Add to graph** to create a plot of the regression equation for the graph specified by the drop-down list. This option on appears if you ran the regression using a graph curve as a data source.
4. Select **95% Confidence and prediction bands** to display confidence and prediction bands on the graph. Confidence and Prediction Bands on page 98

   Note: This option only appears if you select either **Create new graph** or **Add to graph**.
5. Select **Extend fit to axes** to extend the equation curve to intersect the Y-axis.
6. Select **Add equation to graph title** to insert the equation of the curve fit under the title of the graph.
7. After selecting the graphed results you want, click **Finish**.

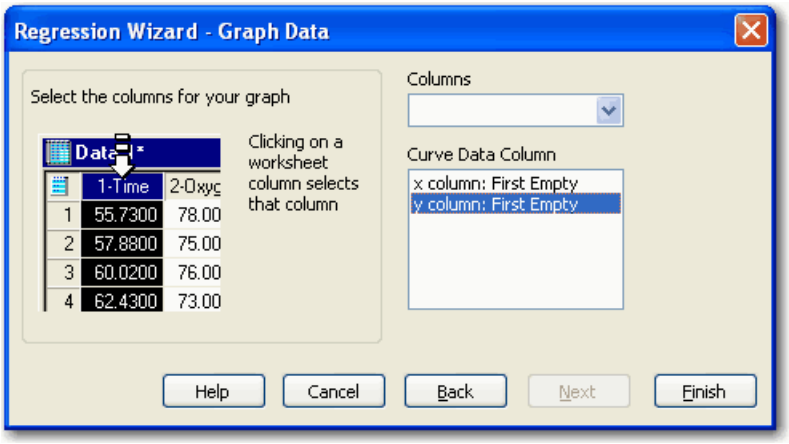
   Click **Next** only if you want to select the specific columns used to contain the data for the fitted curve.

Selecting Columns for Graph Data

1. **To select the specific columns to use for the plotted results,** click the columns in the worksheet where you want the results to always appear.

   Remember: These settings are reused each time you perform a regression and overwrite data if it exists in these columns in subsequent worksheets. To avoid overwriting data, use **First Empty** to place the fitted curve results in empty columns.

   Figure 46: Selecting the graph results columns. These settings are retained between sessions.



2. Click **Finish**.

Finishing the Regression

After clicking **Finish**, all your results are displayed in the worksheet, report, and graph. The initial defaults are to save parameter and computed dependent variable values to the worksheet, to create a statistical report, and to graph the results.

To change the results that are saved, click **Next** to go through the entire wizard, changing your settings as desired.

Setting Nonlinear Regression Report Options

Use the **Report Options for Nonlinear Regression** dialog box to:

- **Set assumption checking options.** Assumption Checking on page 84
- **Specify the residuals to display and save them to the worksheet.** Residuals on page 85
- **Display confidence and prediction intervals and save them to the worksheet.** More Statistics on page 86
- **Display the PRESS Prediction Factor.** PRESS Statistic on page 104
- **Specify tests to identify outlying or influential data points.** Other Diagnostics on page 87
- **Display power.** Other Diagnostics on page 87

Tip: To open the **Report Options for Nonlinear Regression** dialog box, click **Report** on the **Regression Wizard - Numeric Output Options** panel.

Assumption Checking

Select the **Assumption Checking** tab from the **Report Options for Nonlinear Regression** to view the Normality, Constant Variance, and Durbin-Watson options. These options test your data for its suitability for regression analysis by checking three assumptions that a linear regression makes about the data. A nonlinear regression assumes:

- That the source population is normally distributed about the regression.
- The variance of the dependent variable in the source population is constant regardless of the value of the independent variable(s).

- That the residuals are independent of each other.

All assumption checking options are selected by default. Only disable these options if you are certain that the data was sampled from normal populations with constant variance and that the residuals are independent of each other.

Normality Testing. SigmaPlot uses the Kolmogorov-Smirnov test to test for a normally distributed population.

Constant Variance Testing. SigmaPlot tests for constant variance by computing the Spearman rank correlation between the absolute values of the residuals and the observed value of the dependent variable. When this correlation is significant, the constant variance assumption may be violated, and you should consider trying a different model (for example, one that more closely follows the pattern of the data), or transforming one or more of the independent variables to stabilize the variance. User-Defined Transforms on page 19

P Values for Normality and Constant Variance. The P value determines the probability of being incorrect in concluding that the data is not normally distributed (P value is the risk of falsely rejecting the null hypothesis that the data is normally distributed). If the P computed by the test is greater than the P set here, the test passes.

To require a stricter adherence to normality and/or constant variance, increase the P value. Because the parametric statistical methods are relatively robust in terms of detecting violations of the assumptions, the suggested value in SigmaPlot is 0.05. Larger values of P (for example, 0.10) require less evidence to conclude that the residuals are not normally distributed or the constant variance assumption is violated.

To relax the requirement of normality and/or constant variance, decrease P. Requiring smaller values of P to reject the normality assumption means that you are willing to accept greater deviations from the theoretical normal distribution before you flag the data as non-normal. For example, a P value of 0.01 for the normality test requires greater deviations from normality to flag the data as non-normal than a value of 0.05.

Note: Although the assumption tests are robust in detecting data from populations that are non-normal or with non-constant variances, there are extreme conditions of data distribution that these tests cannot detect; however, these conditions should be easily detected by visually examining the data without resorting to the automatic assumption tests.

Durbin-Watson Statistic. SigmaPlot uses the Durbin-Watson statistic to test residuals for their independence of each other. The Durbin-Watson statistic is a measure of serial correlation between the residuals. The residuals are often correlated when the independent variable is time, and the deviation between the observation and the regression line at one time are related to the deviation at the previous time. If the residuals are not correlated, the Durbin-Watson statistic will be 2.

Difference from 2 Value. Enter the acceptable deviation from 2.0 that you consider as evidence of a serial correlation in the Difference for 2.0 box. If the computed Durbin-Watson statistic deviates from 2.0 more than the entered value, SigmaPlot warns you that the residuals may not be independent. The suggested deviation value is 0.50, for example, Durbin-Watson Statistic values greater than 2.5 or less than 1.5 flag the residuals as correlated.

To require a stricter adherence to independence, decrease the acceptable difference from 2.0.

To relax the requirement of independence, increase the acceptable difference from 2.0.

Residuals

Click the **Residuals** tab in the **Report Options for Nonlinear Regression** dialog box to view the Predicted Values, Raw, Standardized, Studentized, Studentized Deleted, and Report Flagged Values Only options.

Studentized Residuals. Studentized residuals scale the standardized residuals by taking into account the greater precision of the regression line near the middle of the data versus the extremes. The Studentized residuals tend to be distributed according to the Student t distribution, so the t distribution can be used to define "large" values of the Studentized residuals. SigmaPlot automatically flags data points with "large" values of the Studentized residuals, for example, outlying data points; the suggested data points flagged lie outside the 95% confidence interval for the regression population.

To include studentized residuals in the report, make sure this check box is selected. Click the selected check box if you do not want to include studentized residuals in the worksheet.

Studentized Deleted Residuals. Studentized deleted residuals are similar to the Studentized residual, except that the residual values are obtained by computing the regression equation without using the data point in question.

To include Studentized deleted residuals in the report, make sure this check box is selected. Click the selected check box if you do not want to include Studentized deleted residuals in the worksheet.

SigmaPlot can automatically flag data points with "large" values of the Studentized deleted residual, for example, outlying data points; the suggested data points flagged lie outside the 95% confidence interval for the regression population.

Note: Both Studentized and Studentized deleted residuals use the same confidence interval setting to determine outlying points.

Raw Residuals. The raw residuals are the differences between the predicted and observed values of the dependent variables. To include raw residuals in the report, make sure this check box is selected. Click the selected check box if you do not want to include raw residuals in the worksheet.

To assign the raw residuals to a worksheet column, select the number of the desired column from the corresponding drop-down list. If you select none from the drop-down list and the Raw check box is selected, the values appear in the report but are not assigned to the worksheet.

Predicted Values. Use this option to calculate the predicted value of the dependent variable for each observed value of the independent variable(s), then save the results to the worksheet. Click the selected check box if you do not want to include raw residuals in the worksheet.

To assign predicted values to a worksheet column, select the worksheet column you want to save the predicted values to from the corresponding drop-down list. If you select none and the Predicted Values check box is selected, the values appear in the report but are not assigned to the worksheet.

Standardized Residuals. The standardized residual is the residual divided by the standard error of the estimate. The standard error of the residuals is essentially the standard deviation of the residuals, and is a measure of variability around the regression line. To include standardized residuals in the report, make sure this check box is selected. Click the selected check box if you do not want to include raw residuals in the worksheet.

Flag Values >. SigmaPlot automatically flags data points lying outside of the confidence interval specified in the corresponding box. These data points are considered to have "large" standardized residuals, for example, outlying data points. You can change which data points are flagged by editing the value in the **Flag Values >** edit box. The suggested residual value is 2.5.

Report Flagged Values Only. To include only the flagged standardized and Studentized deleted residuals in the report, make sure the Report Flagged Values Only check box is selected. Clear this option to include all standardized and Studentized residuals in the report.

More Statistics

Click the **More Statistics** tab in the **Report Options for Nonlinear Regression** dialog box to view options for Confidence and Prediction Intervals and PRESS Prediction Error.

Confidence Intervals. You can set the confidence interval for the population, regression, or both and then save them to the worksheet.

- **Prediction Interval.** The confidence interval for the population gives the range of values that define the region that contains the population from which the observations were drawn. To include confidence intervals for the population in the report, make sure the Population check box is selected. Click the selected check box if you do not want to include the confidence intervals for the population in the report.
- **Confidence Interval.** The confidence interval for the regression line gives the range of values that defines the region containing the true mean relationship between the dependent and independent variables, with the specified level of confidence.

To include confidence intervals for the regression in the report, make sure the Regression check box is selected, then specify a confidence level by entering a value in the percentage box. The confidence level can be any value from 1 to 99. The suggested confidence level for all intervals is 95%.

Click the selected check box if you do not want to include the confidence intervals for the population in the report.
Click the selected check box if you do not want to include the confidence intervals for the population in the report.

- **Saving Confidence Intervals to the Worksheet.** To save the confidence intervals to the worksheet, select the column number of the first column you want to save the intervals to from the Starting in Column drop-down list. The selected intervals are saved to the worksheet starting with the specified column and continuing with successive columns in the worksheet.

PRESS Prediction Error. The PRESS Prediction Error is a measure of how well the regression equation fits the data. Leave this check box selected to evaluate the fit of the equation using the PRESS statistic. Click the selected check box if you do not want to include the PRESS statistic in the report.

AICc -- Akaike Information Criterion. The Akaike Information Criterion provides a method for measuring the relative performance in fitting a regression model to a given set of data.

Other Diagnostics

Click the **Other Diagnostics** tab in the **Report Options for Nonlinear Regression** dialog box to view options Influence, DFFITS, leverage, Cook's Distance and power.

Influence. Influence options automatically detect instances of influential data points. Most influential points are data points which are outliers, that is, they do not do not "line up" with the rest of the data points. These points can have a potentially disproportionately strong influence on the calculation of the regression line. You can use several influence tests to identify and quantify influential points.

- **DFFITS.** DFFITS is the number of estimated standard errors that the predicted value changes for the ith data point when it is removed from the data set. It is another measure of the influence of a data point on the prediction used to compute the regression coefficients.

Predicted values that change by more than two standard errors when the data point is removed are considered to be influential.

Select **DFFITS** to compute this value for all points and flag influential points, for example those with DFFITS greater than the value specified in the Flag Values > edit box. The suggested value is 2.0 standard errors, which indicates that the point has a strong influence on the data. To avoid flagging more influential points, increase this value; to flag less influential points, decrease this value.

- **Leverage.** Leverage is used to identify the potential influence of a point on the results of the regression equation. Leverage depends only on the value of the independent variable(s). Observations with high leverage tend to be at the extremes of the independent variables, where small changes in the independent variables can have large effects on the predicted values of the dependent variable.

Select Leverage to compute the leverage for each point and automatically flag potentially influential points, for example, those points that could have leverages greater than the specified value times the expected leverage. The suggested value is 2.0 times the expected leverage for the regression. To avoid flagging more potentially influential points, increase this value; to flag points with less potential influence, lower this value.

- **Cook's Distance.** Cook's distance is a measure of how great an effect each point has on the estimates of the parameters in the regression equation. Cook's distance assesses how much the values of the regression coefficients change if a point is deleted from the analysis. Cook's distance depends on both the values of the independent and dependent variables.

Select **Cook's Distance** to compute this value for all points and flag influential points, for example, those with a Cook's distance greater than the specified value. The suggested value is 4.0. Cook's distances above 1 indicate that a point is possibly influential. Cook's distances exceeding 4 indicate that the point has a major effect on the values of the parameter estimates. To avoid flagging more influential points, increase this value: to flag less influential points, lower this value.

Power. The power of a regression is the power to detect the observed relationship in the data. The alpha is the acceptable probability of incorrectly concluding there is a relationship.

Select **Power** to compute the power for the linear regression data. Change the alpha value by editing the number in the **Alpha Value** edit box. The suggested value is α = 0.05. This indicates that a one in twenty chance of error is acceptable, or that you are willing to conclude there is a significant relationship when P < 0.05.

Report Flagged Values Only. To only include only the influential points flagged by the influential point tests in the report, select **Report Flagged Values Only**. Clear this option to include all influential points in the report.

Running a Regression From a Notebook

Because regression equations can be treated like any other notebook item, you can select and open regression equations directly from a notebook. This is particularly convenient if you have created or stored equations along with the rest of your graphs and data.

1. In the **Notebook Manager**, view the notebook with the equation you want to use, and double-click the equation. The **Regression Wizard** appears with the equation selected.
2. Select the variables as prompted by clicking a curve or worksheet columns. Note that at this point you can open and view any notebook, worksheet or page you would like, and pick your variables from that source.

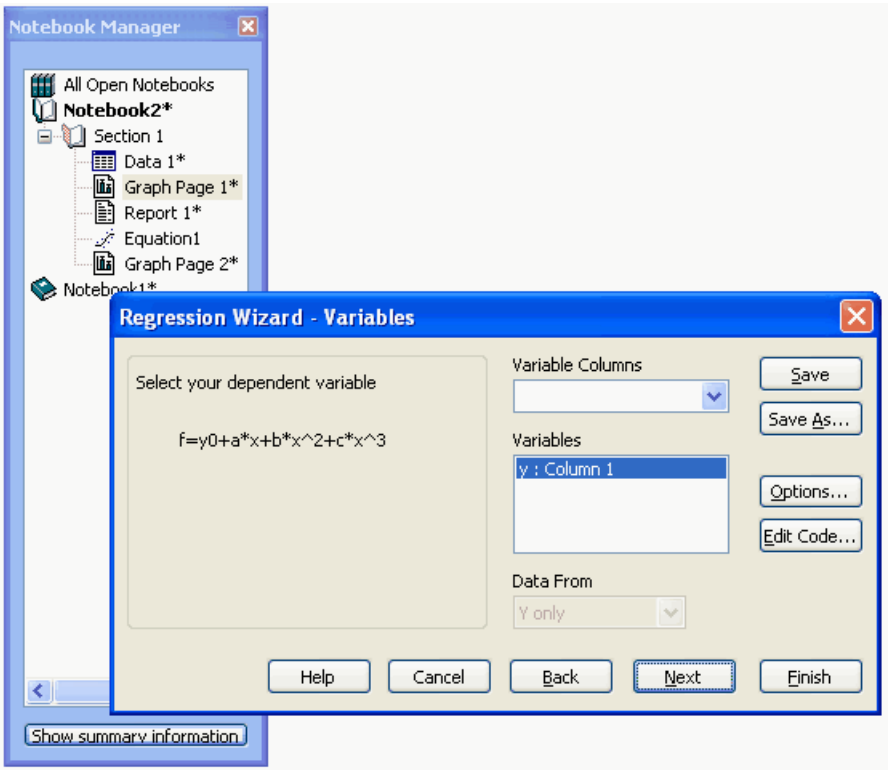


Figure 47: Selecting a regression equation from a Notebook to Start the Regression Wizard

3. Click **Finish** to complete the regression, or click **Next** if you want to view initial results or change your results options.

Creating New Regression Equations

You can create new equations by using the **Function** dialog box. Here you can set the equations, variables, initial parameters, constraints and other options. You can create new regression equations two different ways:

- On either the **Regression Wizard**, the **Dynamic Fit Wizard**, or the **Global Fit Wizard**, click **New** or **Edit Code**, or
- Right-click in a notebook section, and on the shortcut menu, click **Equation**.

When you create a new equation, the **Function** dialog box appears with blank headings.

Viewing and Editing Code

To view the code for the current equation document, click**Edit Code** in the **Regression Wizard**, **Dynamic Fit Wizard**, or **Global Fit Wizard**. Editing Code on page 129

You can click the **Edit Code** button from the equation or variables panels. The **Edit Code** button opens the **Function** dialog box. All settings for the equation are displayed.

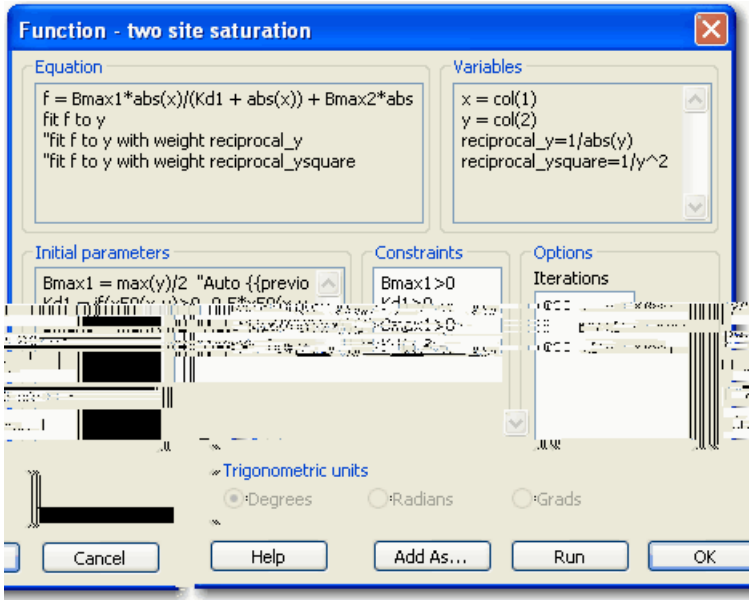


Figure 48: Viewing the code for a built-in equation in the Function dialog box.

Tip: You cannot edit the Equations, Parameters, and Variables for built-in SigmaPlot equations; however, you can edit and save built-in equations as new equations. Click **Add As**, add the equation to the desired section, and then edit the Equations, Variables and Parameters as desired.

You can also copy and paste equations from notebook to notebook like any other notebook item. You can also edit pasted built-in equations. Editing Code on page 129

Saving Regression Equation Changes

When you edit an equation using the **Equation Options** or **Function** dialog boxes, or when you add an equation, all changes are updated to the equation in the library or notebook. However, just like other notebook items, these changes are not saved to the file until the notebook is saved. Changes made to regression libraries are automatically saved when you close the Regression, Dynamic Fit, or Global Fit Wizard.

You can also save changes to regression libraries using the **Save** or **Save As** buttons in the **Regression Wizard**. This saves the current regression library notebook to disk. **Save As** allows you to save the regression library to a new file.

If you have a regression library open as a notebook, you can also save changes by saving the notebook. To save the notebook, press **Ctrl + S**.

Variable Options

Data Format Options. If you use data columns from the worksheet, you can specify the data format to use in the variables panel of the Regression Wizard. By default, the data format when assigning columns from the worksheet is XY Pair.

The data format options are:

- **XY pair.** Select an x and a y variable.
- **Y only.** Select only a y variable column.
- **XY column means.** Pick one x column, then multiple y columns; the y columns will be graphed as means.
- **Y column means only.** Pick multiple y columns; the columns will be graphed as means.
- **From Code.** Uses the current settings as shown when editing code.
- **XY Replicate.** Select X and multiple Y columns. Rows of the Y columns are replicate measurements.
- **Y Replicates.** Select multiple Y columns. Rows of the Y column are replicate measurements.

When you use an existing graph as your data source, the Regression Wizard displays a format reflecting the data format of the graph. You cannot change this format unless you switch to using the worksheet as your data source, or run the regression directly from editing the code.

Multiple Independent Variables

Although the Standard Regression Library only supports up to two independent variables, the curve fitter can accept up to ten. To use models that have more than two independent variables, simply create or open a model with the desired equation and variables. The Regression Wizard prompts to select columns for each defined variable

Equation Options

If the curve fitter fails to find a good fit for the curve, you can try changing the equation options to see if you can improve the fit. To set options for a regression, click the **Options** button in the **Variables** panel of the **Regression**, **Dynamic Fit**, or **Global Fit Wizard**.

Tip: If you want to edit the settings in the equation document manually, click the **Edit Code** button. Editing Code on page 129

Use the **Equation Options** dialog box to:

- Change initial parameter values.
- Add or change constraints.
- Change constant values.
- Use weighted fitting, if it is available.
- Change convergence options.

Parameters

The default setting for the initial parameter value is shown **Automatic**. The **Automatic** setting available with the built-in SigmaPlot equations uses algorithms that analyze your data to predict initial parameter estimates. These do not work in all cases, so you may need to enter a different value. Just click the parameter you want to change, and make the change in the edit box.

The values that appear in the **Initial Parameters** drop-down list were previously entered as parameter values. Any parameter values you enter will also be retained between sessions.

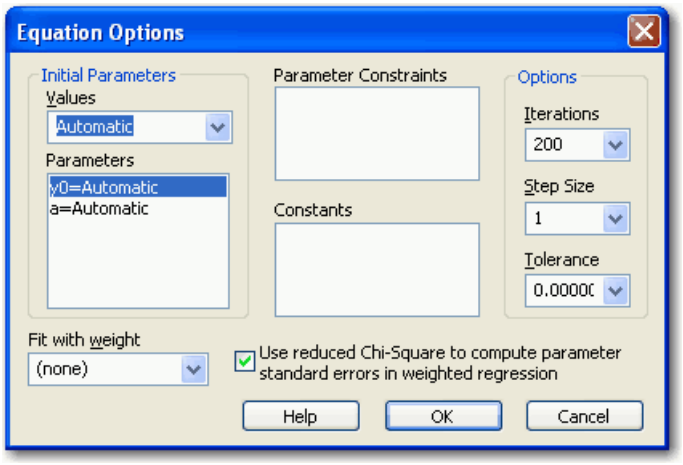


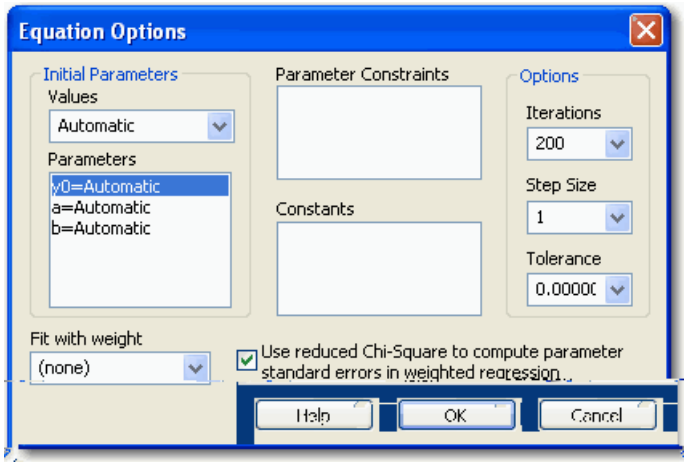
Figure 49: Selecting Numeric Initial Parameters in the Equation Options dialog box.

Parameters can be either a numeric value or a function. The value of the parameter should approximate the final result, in order to help the curve fitter reach a valid result, but this depends on the complexity and number of parameters of the equation. Often an initial parameter nowhere near the final result will still work. However, a good initial estimate helps guarantee better and faster results.

Constraints

Use *constraints* to set limits and conditions for parameter values, restricting the regression search range and improving curve fitter speed and accuracy. Constraints are often unnecessary, but should always be used whenever appropriate for your model.

Constraints are also useful to prevent the curve fitter from testing unrealistic parameter values. For example, if you know that a parameter should always be negative, you can enter a constraint defining the parameter to be always less than 0.



You can also use constraints if the regression produces parameter values that you know are inaccurate. Simply click **Back** from the initial results panel, click the **Options** button, and enter constraint(s) that prevent the wrong parameter results.

Note that a parameter equals a constraint value at the completion of the fit, the constraint is called *active*. You can view these constraints from the initial results panel by clicking **View Constraints**. Checking Use of Constraints on page 96

Entering Parameter Constraints

To enter constraints, click the **Constraints** edit box, and type the desired constraint(s), using the transform language operators.

A constraint must be a linear equation of the equation parameters, using an equal (=) or inequality (< or >) sign. For example, you could enter the following constraints for the parameters *a, b, c, d,* and *e*:

```
a<1
10*b+c/20>2
d-e=15
a>b+c+d+e
```

However, the constraint

```
a*x<1
```

is illegal, since *x* is a variable, not a parameter, and the constraints

```
b+ c^2> 4
```

```
d*e=1
```

are illegal because they are nonlinear. Inconsistent and conflicting constraints are automatically rejected by the curve fitter.

Defining Constants

Constants that appear in the **Constants** edit window have been previously defined as a constant, rather than a parameter to be determined by the regression. To edit a constant value, or define new constant values, click **Edit Code** on the **Regression Wizard**, **Dynamic Fit Wizard** or **Global Fit Wizard** dialog box.

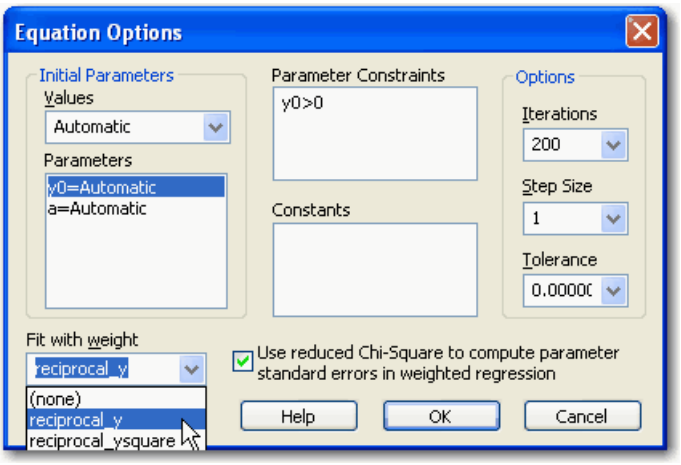
Constants are defined when an equation is created. Currently, you can only define new constants by editing the regression equation code. However, you can redefine any existing constants.

Change only the value of the constant. Do not add new constant values; constant variables must exist in the equation and not be defined already under variables or parameters, so they can only be defined within the code of an equation.

Fit with Weight

You can select from any of the weights listed. Some built-in equations have some predefined values, although most do not. If no weighting options are available for your equation, only the **None** option will be available.

Weighting options appear in the **Fit with Weight** drop-down list. By default, the weighting applied to the fit is **(none)**. To apply a different weighting setting, select a weighting option from the drop-down list.



Weight variables must be defined by editing the regression code.

Iterations

The **Iterations** option sets the maximum number of repeated fit attempts before failure. Each iteration of the curve fitter is an attempt to find the parameters that best fit the model. With each iteration, the curve fitter varies the parameter values incrementally, and tests the fit of that model to your data. When the improvement in the fit from one iteration to the next is smaller than the setting determined by the Tolerance option, the curve fitter stops and displays the results.

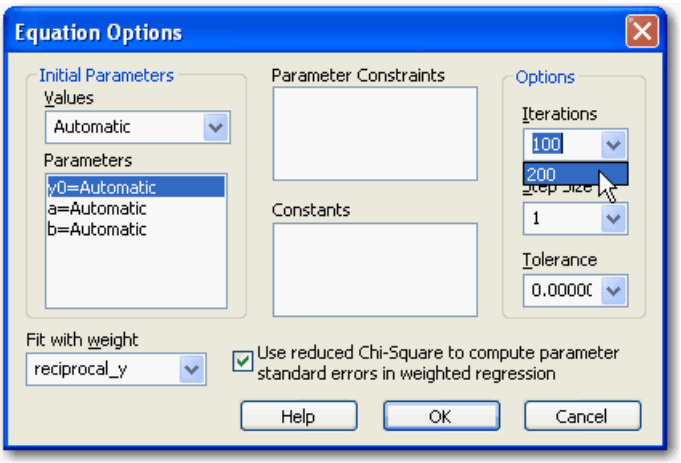


Figure 50: Changing iterations

Change the number of iterations to speed up or improve the regression process, especially if a complex fit requires more than the default of 100 iterations. You can also reduce the number of iterations if you want to end a fit to check on its interim progress before it takes too many iterations.

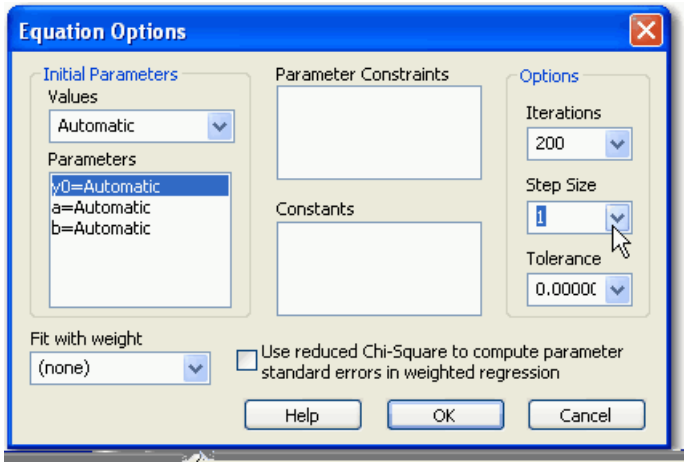
Setting **Iterations** to 0 will simply evaluate the dependent variable of the fit equation using the initial parameter values.

To change the maximum number of iterations, enter the number of iterations to use, or select a previously used number of iterations from the drop down list.

When the maximum number of iterations is reached, the regression stops and the current results are displayed in the initial parameters panel. If you want to continue with more iterations, you can click **More Iterations** on the **Regression Wizard**.

Step Size

Step size, or the limit of the initial change in parameter values used by the curve fitter as it tries, or iterates, different parameter values, is a setting that can be changed to speed up or improve the regression process.



A large step size can cause the curve fitter to wander too far away from the best parameter values, whereas a step size that is too small will result in slow convergence to the best parameters.

For most functions, the default step size value is 1. To change the **Step Size** value, type the desired step size in the **Step Size** edit box, or select a previously defined value from the drop-down list.

Tolerance

The **Tolerance** option controls the condition that must be met in order to end the regression process. When the absolute value of the difference between the sum of squares of the residuals (square root of the sum of squares of the residuals), from one iteration to the next, is less than the tolerance value, the iteration stops.

When the tolerance condition has been met, a minimum of the sum of squares has usually been found, which indicates a correct solution. However, local minima in the sum of squares can also cause the curve fitter to find an incorrect solution.

Decreasing the value of the tolerance makes the requirement for finding an acceptable solution more strict; increasing the tolerance relaxes this requirement.

The default tolerance setting is1e-10. To change the tolerance value, type the desired value in the **Tolerance** edit box, or select a previously defined value from the drop-down list.

Watching The Fit Progress

During the regression process, the Regression fit progress dialog box displays the number of iterations completed, the norm value for each iteration, and a progress bar indicating the percent complete of the maximum iterations.

Cancelling a Regression

To stop a regression while it is running, click **Cancel**. The initial results appear, displaying the most recent parameter values, and the sum of square value. You can continue the regression process by clicking **More Iterations** on the **Regression Wizard**.

Interpreting Fit Results

When you click **Next** from the variables panel, the regression process completes by either converging, reaching the maximum number of iterations, or encountering an error. When any of these conditions are met, or whenever there is an error or warning, the initial results panel is displayed.

Completion Status Messages

A message displaying the condition under which the regression completed is displayed in the upper left corner of the Regression Wizard. If the regression completed with convergence, the message:

```
Converged, tolerance satisfied
```

is displayed.

Otherwise, another status or error message is displayed. Regression Results Messages on page 112

Rsqr

R^2 is the *coefficient of determination*, the most common measure of how well a regression model describes the data. The closer R^2 is to one, the better the independent variables predict the dependent variable.

R^2 equals 0 when the values of the independent variable does not allow any prediction of the dependent variables, and equals 1 when you can perfectly predict the dependent variables from the independent variables.

Fit Results

The initial results are displayed in the results window, in five columns.

- **Parameter.** The parameter names are shown in the first column. These parameters are derived from the original equation.
- **Value.** The calculated parameter values are shown in the second column.
- **StdErr.** The asymptotic standard errors of the parameters are displayed in column three. The standard errors and coefficients of variation can be used as a gauge of the fitted curve's accuracy.
- **CV(%).** The parameter coefficients of variation, expressed as a percentage, are displayed in column four. This is the normalized version of the standard errors:

$$CV = \frac{standard\ error \times 100}{paramater\ value}$$

The coefficient of variation values and standard errors can be used as a gauge of the accuracy of the fitted curve.

- **Dependency.** The last column shows the parameter dependencies. The dependence of a parameter is defined to be

$$dependence = 1 - \frac{variance\ of\ the\ parameter,\ other\ parameters\ constant}{variance\ of\ the\ parameter,\ other\ parameters\ changing}$$

Parameters with dependencies near 1 are strongly dependent on one another. This may indicate that the equation(s) used are too complicated and *over-parameterized*—too many parameters are being used, and using a model with fewer parameters may be better.

Changing the Regression Equation or Variables

To go back to any of the previous panels, click **Back**. This is especially useful if you need to change the model (equation) used, or if you need to modify any of the equation options and try the curve fit again.

More Iterations

If the maximum number of iterations was reached before convergence, or if you canceled the regression, the **More Iterations** button is available. Click **More Iterations** to continue fotimttontnrtn csoti pndoeind

To place any of these values in a column in the worksheet, simply check the results you want to keep. If you want to set a specific column in which to always place these values, you can click a column on a worksheet for each result.

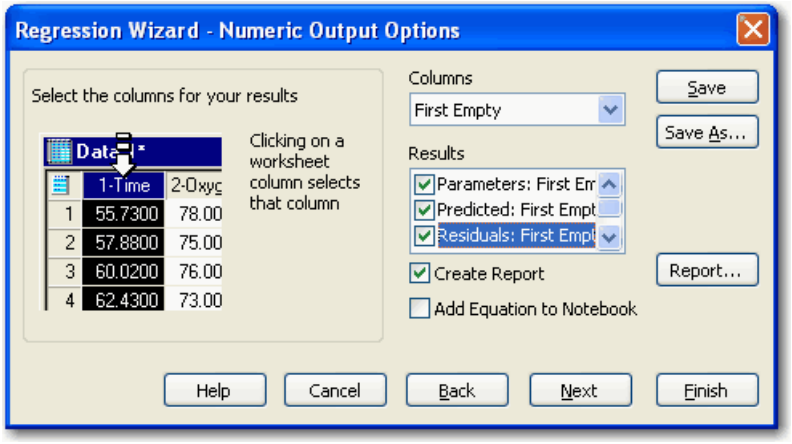


Figure 51: Generating and Saving a Report from the Regression Wizard

Create Report. Select to save regression reports to the current notebook section.

Adding Equation to Notebook. To add the current regression equation to the current notebook, select Add Equation to Notebook. If this option is selected, a copy of the equation is added to the current section of your notebook.

Graphing Regression Equations

SigmaPlot graphs the results of a regression as a fitted curve and creates a curve or graph by default. If you want to disable graphed results, you can change the options in the **Regression Wizard - Graph Options** panel. Note that SigmaPlot retains these settings from session to session.

From the graph panel, you can choose to plot the results either by:

- Adding a plot to an existing graph. This option is only available if the fitted variables were assigned by selecting them from a graph.
- Creating a new graph of the original data and fitted curve.

To add a plot to an existing graph, select **Add Curve to**, then select the graph to which you want to add a plot from the drop-down list. The drop-down list includes all the graphs on the current page. If there is no existing graph, this option is dimmed.

If you want to specify the columns used to plot the fitted curve, click **Next**. Otherwise, the data is placed in the first available columns.

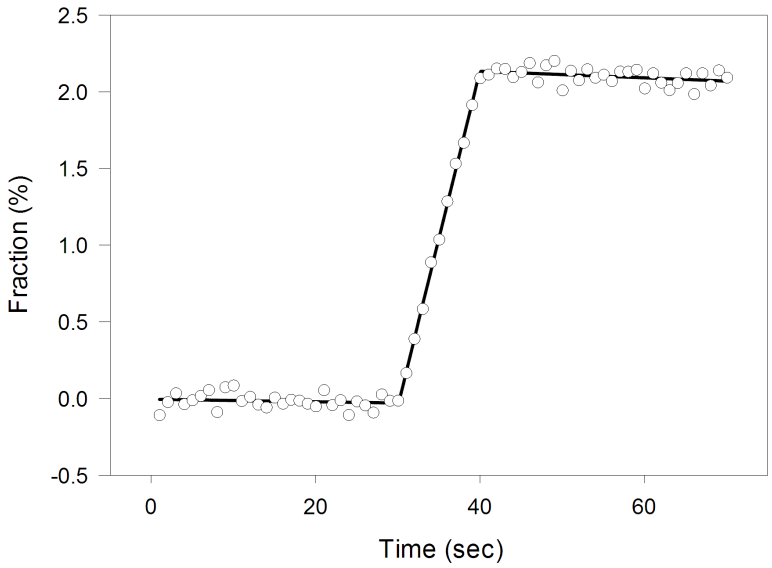


Figure 52: A Fitted Curve Added to the Graph

To create a new graph, select **Create New Graph**. Click **Finish** to create a new notebook section containing a worksheet of the plotted data and graph page.

Data Plotted for Regression Curves

You can specify the worksheet columns used to add a fitted curve to an existing graph, or to create a new graph, by clicking **Next** from the graph panel.

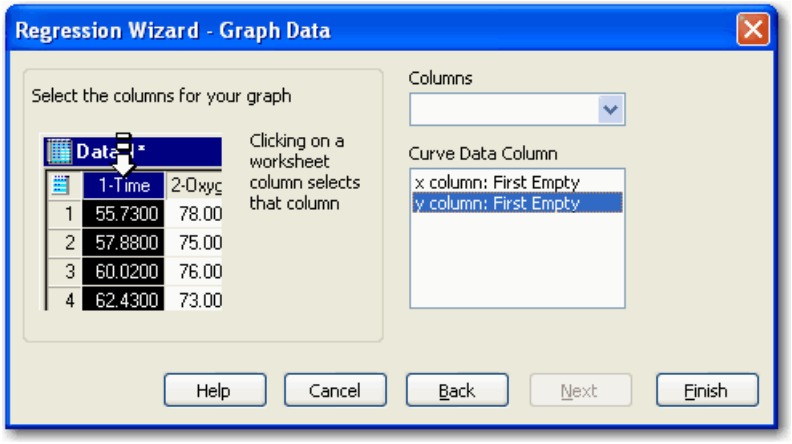


Figure 53: The Regression Wizard Pick Output Dialog Box

From this panel you can select worksheet columns for X, Y, (and Z data for 3D graphs) by clicking worksheet columns. The default of First Empty places the results in the first available column after the last filled cell.

Confidence and Prediction Bands

The term *confidence band* refers to the region of uncertainties in the predicted values over a range of values for the independent variable. The term *prediction band* refers to the region of uncertainties in predicting the response for a single additional observation at each point within a range of independent variable values. Prediction bands are always wider than confidence bands. Each band appears on the graph as a multiple line plot with two curves. One represents

the upper limits of the confidence intervals and the other represents the lower limits of the confidence intervals. The independent variable values used to compute the confidence bands are the same values used to create the fit curve.

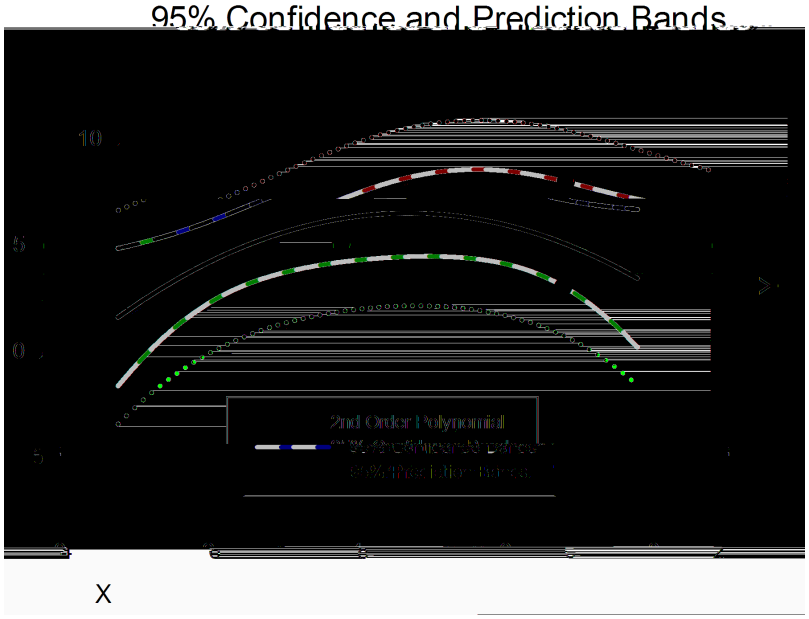


Figure 54: A 2D Graph with Confidence and Prediction Bands

Confidence and Prediction Band Computational Formulas

Calculation of the limits of both bands is based upon a quantity that generalizes the notion of leverage at a data point. At a given value *x* of the independent variable, define

$$c = (gradF)^T \cdot Cov \cdot (gradF)$$

where grad*F* is the (parameter) gradient of the model *F*, evaluated at *x* and at the best-fit parameter values, and *Cov* is the covariance matrix computed at the final iteration of the regression.

After computing *c*, the upper and lower limits of both bands are given by:

$$Confidence\ Band = y(x) \pm \sqrt{c}\sqrt{MS_{res}}\, CriticalT(conf.levels\%, DF)$$

$$Prediction\ Band = y(x) \pm \sqrt{c+1}\sqrt{MS_{res}}\, CriticalT(conf.levels\%, DF)$$

where $y(x)$ is the predicted value at *x*, MS_{res} is the mean residual sum of squares, *conf.level%* is the percent confidence level, *DF* is the residual degree of freedom (number of data points - number of parameters), and CriticalT(*conf.level %,DF*) is the inverse *T*-Distribution with *DF* degrees of freedom evaluated 1-α/2, where α satisfies *conf.level%* = 100(1-α)%.

Changing Confidence and Prediction Band Percentage Values

You add confidence and prediction bands on the **Graph Options** panel of the **Regression Wizard**. This value should agree with any confidence interval results in a report.

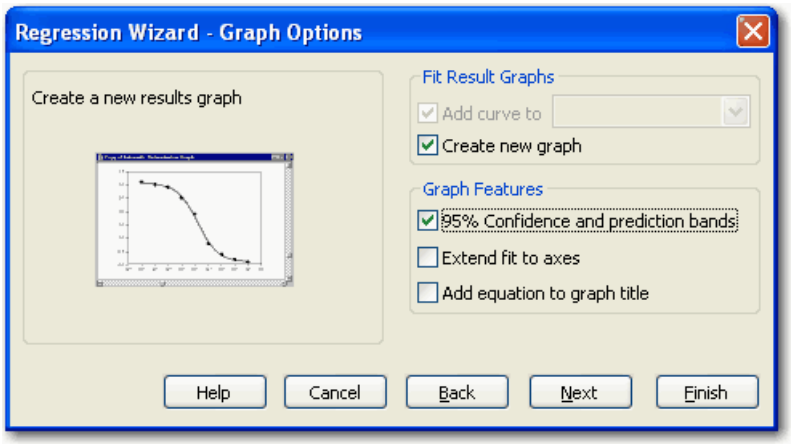


Figure 55: Selecting to Add Confidence and Prediction Bands to a graph

While the default confidence level for confidence and prediction bands is 95%, you can change this value in the **Report Options for Nonlinear Regression** dialog box.

1. Click **Back** if you are viewing the **Regression Wizard - Graph Options** dialog box to the **Numeric Output Options** panel.

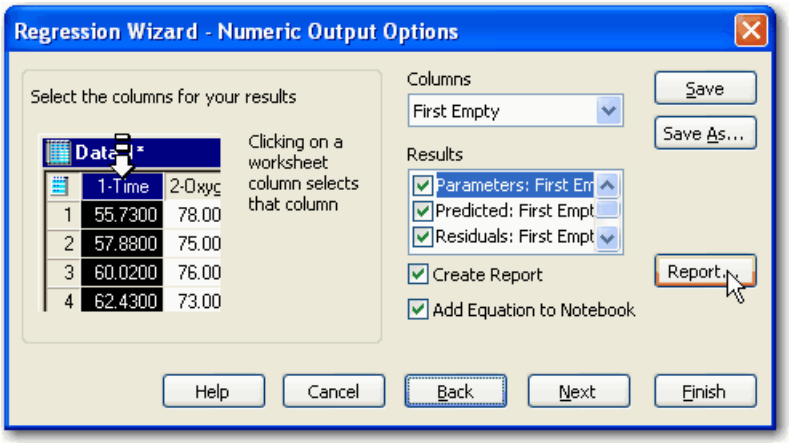


Figure 56: Click Report to open the Options for Nonlinear Regression dialog box.

2. Click **Report** to open the **Options for Nonlinear Regressions** dialog box. Here is where you control what you would like to appear in the report.

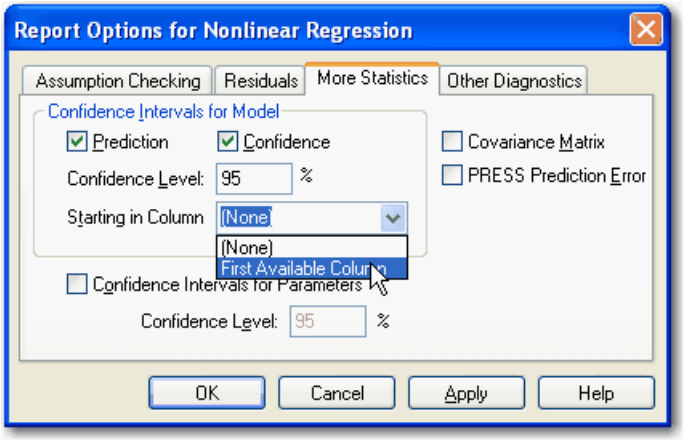


Figure 57: Selecting the Report Options

3. Click the **More Statistics** tab.

4. Under **Confidence Intervals**, select the **Prediction** and **Confidence** intervals you would like to appear in the report.
5. Set the percentage for the confidence level in the **Confidence Level** box.
6. In the **Starting in Column** drop-down list, select **First Available Column** or select **(None)** if you don't want the confidence and prediction intervals to appear in the worksheet,
7. Click **OK** to save the changes and close the dialog box.
8. Click **Next** in the **Regression Wizard**.

   The **Regression Wizard - Graph Options** panel appears with the new percentage for the confidence and prediction bands.

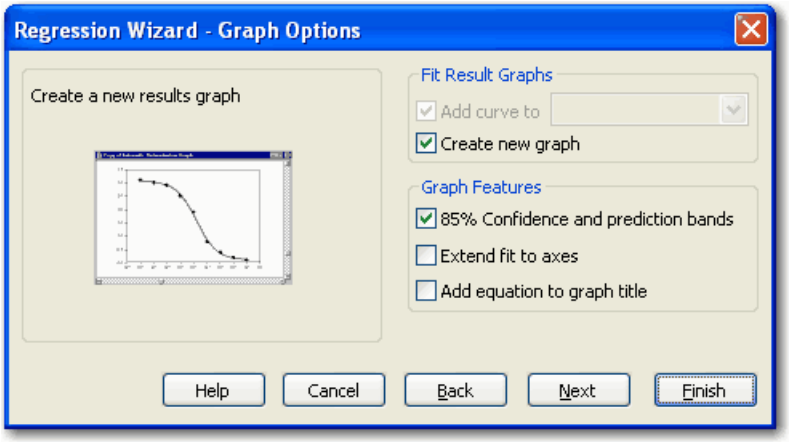


Figure 58: Selecting Confidence and Prediction Bands

When you click **Finish** in the Regression Wizard, the confidence and prediction bands appear on the graph. If you selected **First Available Column** in the **More Statistics** tab of the **Report Options for Nonlinear Regression** dialog box, beginning in the first empty column, four columns of graph data appear in the worksheet which represent the upper and lower limits of the confidence and prediction bands.

Data 28*

	11-85% ConfBand-L	12-85% ConfBand-U	13-85% PredBand-L	14-85% PredBand-U
1	2.5729	4.3949	1.3268	5.6410
2	2.5708	4.3845	1.3222	5.6330
3	2.5686	4.3741	1.3177	5.6250
4	2.5663	4.3638	1.3131	5.6171
5	2.5640	4.3536	1.3085	5.6092
6	2.5616	4.3435	1.3038	5.6013
7	2.5592	4.3334	1.2991	5.5934
8	2.5567	4.3234	1.2944	5.5856
9	2.5541	4.3134	1.2897	5.5778
10	2.5514	4.3035	1.2849	5.5701
11	2.5487	4.2937	1.2800	5.5624

Figure 59: Worksheet Columns Representing Data for the Confidence and Prediction Bands

These results also appear in the report. For more information, see Confidence and Prediction Intervals on page 106.

Interpreting Regression Reports

The Regression Wizard can automatically generate reports for each curve fitting session. The statistical results are displayed to four decimal places of precision by default.

Reports are displayed using the SigmaPlot report editor. Using the Report Editor

Equation Code

This is a printout of the code used to generate the regression results. For more information, see Editing Code on page 129.

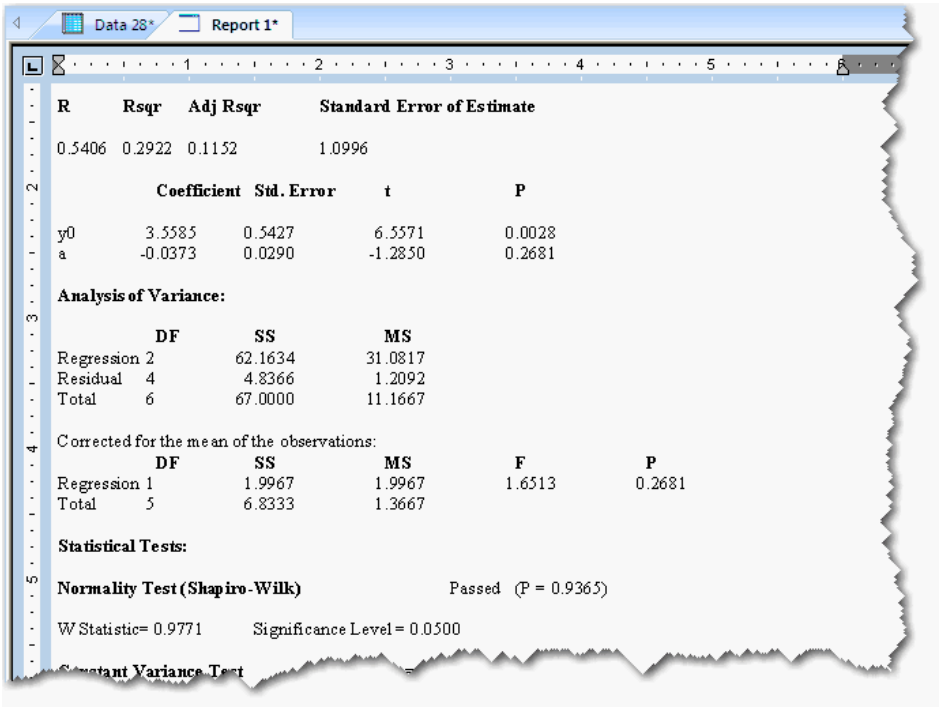


Figure 60: Regression Report

R and R Squared

The *multiple correlation coefficient*, *R*, and the coefficient of determination, R^2, are both measures of how well the regression model describes the data. *R* values near 1 indicate that the equation is a good description of the relation between the independent and dependent variables. *R* equals 0 when the values of the independent variable does not allow any prediction of the dependent variables, and equals 1 when you can perfectly predict the dependent variables from the independent variables R^2

Adjusted R Squared

The adjusted R^2, R^2_{adj}, is also a measure of how well the regression model describes the data, but takes into account the number of independent variables, which reflects the degrees of freedom. Larger R^2_{adj} values (nearer to 1) indicate that the equation is a good description of the relation between the independent and dependent variables.

Standard Error of the Estimate

The standard error of the estimate $S_{y|x}$ is a measure of regression plane of the actual variability about the regression plane of the underlying population. The underlying population generally falls within about two standard errors of the observed sample.

Statistical Summary Table

The standard error, *t* and *P* values are approximations computed at the final iteration of the regression.

Estimate. The value for the constant and coefficients of the independent variables for the regression model are listed.

Standard Error. The asymptotic standard errors measure the uncertainties in the estimates of the regression coefficients (analogous to the standard error of the mean). The true regression coefficients of the underlying population are generally within about two standard errors of the predicted coefficients. Large standard errors may indicate multicollinearity. The default procedure for computing standard errors is based on whether or not the regression problem is weighted. In an unweighted problem, the standard error for each parameter includes a factor that estimates the standard deviation of the observed data. In this case, it is assumed that the errors for all data points have the same variance.

In a weighted problem, there are two options for computing standard errors. One option includes the factor that estimates the standard deviation of the observed data. This option is called *reduced chi-square*. The other option omits this factor in the computation. To select the option for standard errors, go to the Equation Options dialog box.

t statistic. The *t* statistic tests the null hypothesis that the coefficient of the independent variable is zero, that is, the independent variable does not contribute to predicting the dependent variable. *t* is the ratio of the regression coefficient to its standard error, or

$$t = \frac{regression\ coefficient}{standard\ error\ of\ regression\ coefficient}$$

You can conclude from *large t* values that the independent variable can be used to predict the dependent variable (for example., that the coefficient is not zero).

P value. *P* is the *P* value calculated for *t*. The *P* value is the probability of being wrong in concluding that the coefficient is not zero (for example, the probability of falsely rejecting the null hypothesis, or committing a Type I error, based on t). The smaller the *P* value, the greater the probability that the coefficient is not zero.

Traditionally, you can conclude that the independent variable can be used to predict the dependent variable when $P < 0.05$.

Analysis of Variance (ANOVA) Table

The ANOVA (analysis of variance) table lists the ANOVA statistics for the regression and the corresponding F value for each step.

SS (Sum of Squares). The sum of squares are measures of variability of the dependent variable.

- The sum of squares due to regression measures the difference of the regression plane from the mean of the dependent variable.
- The residual sum of squares is a measure of the size of the residuals, which are the differences between the observed values of the dependent variable and the values predicted by the regression model.

DF (Degrees of Freedom). Degrees of freedom represent the number of observations and variables in the regression equation.

- The regression degrees of freedom is a measure of the number of independent variables.
- The residual degrees of freedom is a measure of the number of observations less the number of parameters in the equation.

MS (Mean Square). The mean square provides two estimates of the population variances. Comparing these variance estimates is the basis of analysis of variance.

The mean square regression is a measure of the variation of the regression from the mean of the dependent variable, or

$$\frac{sum\ of\ squares\ due\ to\ regression}{regression\ degrees\ of\ freedom} = \frac{SS_{res}}{DF_{res}} = MS_{res}$$

The residual mean square is a measure of the variation of the residuals about the regression plane, or

$$\frac{sum\ of\ squares\ due\ to\ regression}{regression\ degrees\ of\ freedom} = \frac{SS_{res}}{DF_{res}} = MS_{res}$$

The residual mean square is also equal to $S^2_{y\cdot x}$.

F statistic

The *F* test statistic gauges the contribution of the independent variables in predicting the dependent variable. It is the ratio

$$\frac{\textit{regression variation from the dependent variable mean}}{\textit{residual variation about the regression}} = \frac{MS_{reg}}{MS_{res}} = F$$

If *F* is a large number, you can conclude that the independent variables contribute to the prediction of the dependent variable (for example, at least one of the coefficients is different from zero, and the *unexplained variability* is smaller than what is expected from random sampling variability of the dependent variable about its mean). If the *F* ratio is around 1, you can conclude that there is no association between the variables (for example, the data is consistent with the null hypothesis that all the samples are just randomly distributed).

P value

The *P* value is the probability of being wrong in concluding that there is an association between the dependent and independent variables (for example, the probability of falsely rejecting the null hypothesis, or committing a Type I error, based on *F*). The smaller the *P* value, the greater the probability that there is an association.

Traditionally, you can conclude that the independent variable can be used to predict the dependent variable when $P < 0.05$.

PRESS Statistic

PRESS, the *Predicted Residual Error Sum of Squares*, is a gauge of how well a regression model predicts new data. The smaller the PRESS statistic, the better the predictive ability of the model.

The PRESS statistic is computed by summing the squares of the prediction errors (the differences between predicted and observed values) for each observation, with that point deleted from the computation of the regression equation.

Durbin-Watson Statistic

The Durbin-Watson statistic is a measure of correlation between the residuals. If the residuals are not correlated, the Durbin-Watson statistic will be 2; the more this value differs from 2, the greater the likelihood that the residuals are correlated.

Regression assumes that the residuals are independent of each other; the Durbin-Watson test is used to check this assumption. If the Durbin-Watson value deviates from 2 by more than 0.50, a warning appears in the report, for example, if the Durbin-Watson statistic is below 1.50 or above 2.50.

Normality Test

The normality test results display whether the data passed or failed the test of the assumption that the source population is normally distributed around the regression, and the P value calculated by the test. All regressions assume a source population to be normally distributed about the regression line. If the normality test fails, a warning appears in the report.

Failure of the normality test can indicate the presence of outlying influential points or an incorrect regression model.

PRESS = 0.1738

Durbin-Watson Statistic = 2.4780

Normality Test: Passed (P = 0.4760)

Constant Variance Test: Passed (P = 0.5515)

Power of performed test with alpha = 0.0500: 1.0000

Regression Diagnostics:

Row	Predicted	Residual	Std. Res.	Stud. Res.	Stud. Del. Res.
1	-0.0004	-0.1030	-2.1326	-2.2823	-2.3626
2	-0.0012	-0.0173	-0.3587	-0.3812	-0.3786
3	0.0031	0.0403	0.8358	0.8824	0.8808
4	-0.0029	-0.0029	-0.0618	-0.0647	-0.0644
5	-0.0037	-0.0029	-0.0592	-0.0618	-0.0613
6	-0.0045	0.0260	0.5379	0.5589	0.5558

Figure 61: Regression Report Showing Normality Test Results

Constant Variance Test

The constant variance test results displays whether or not the data passed or failed the test of the assumption that the variance of the dependent variable in the source population is constant regardless of the value of the independent variable, and the *P* value calculated by the test. When the constant variance test fails, a warning appears in the report.

If the constant variance test fails, you should consider trying a different model (for example, one that more closely follows the pattern of the data) using a weighted regression, or transforming the independent variable to stabilize the variance and obtain more accurate estimates of the parameters in the regression equation.

If you perform a weighted regression, the normality and equal variance tests use the weighted residuals $w_i(y_i-y_i)$ instead of the raw residuals y_i-y_i.

Power

The power, or sensitivity, of a regression is the probability that the model correctly describes the relationship of the variables, if there is a relationship.

Regression power is affected by the number of observations, the chance of erroneously reporting a difference α (alpha), and the slope of the regression.

Alpha

Alpha (α) is the acceptable probability of incorrectly concluding that the model is correct. An α error is also called a *Type I error* (a *Type I* error is when you reject the hypothesis of no association when this hypothesis is true).

Smaller values of α result in stricter requirements before concluding the model is correct, but a greater possibility of concluding the model is incorrect when it is really correct (a *Type II error*). Larger values of α make it easier to conclude that the model is correct, but also increase the risk of accepting an incorrect model (a Type I error).

Regression Diagnostics

The regression diagnostic results display the values for the predicted values, residuals, and other diagnostic results.

Row. This is the row number of the observation.

Predicted Values. This is the value for the dependent variable predicted by the regression model for each observation.

Residuals. These are the unweighted raw residuals, the difference between the observed and predicted values for the dependent variables.

Standardized Residuals. The standardized residual is the raw residual divided by the standard error of the estimate $S_{y|x}$.

If the residuals are normally distributed about the regression, about 66% of the standardized residuals have values between -1 and +1, and about 95% of the standardized residuals have values between -2 and +2. A larger standardized

residual indicates that the point is far from the regression. Values less than -2.5 or larger than 2.5 may indicate outlying cases.

Studentized Residuals. The Studentized residual is a standardized residual that also takes into account the greater confidence of the predicted values of the dependent variable in the *middle* of the data set. By weighting the values of the residuals of the extreme data points (those with the lowest and highest independent variable values), the Studentized residual is more sensitive than the standardized residual in detecting outliers. This residual is also known as the internally Studentized residual, because the standard error of the estimate is computed using all data.

Studentized Deleted Residuals. The Studentized deleted residual, or externally Studentized residual, is a Studentized residual which uses the standard error of the estimate $S_{y|x(-i)}$, computed after deleting the data point associated with the residual. This reflects the greater effect of outlying points by deleting the data point from the variance computation.

The Studentized deleted residual is more sensitive than the Studentized residual in detecting outliers, since the Studentized deleted residual results in much larger values for outliers than the Studentized residual.

Influence Diagnostics

Row. This is the row number of the observation.

Cook's Distance. Cook's distance is a measure of how great an effect each point has on the estimates of the parameters in the regression equation. It is a measure of how much the values of the regression coefficients would change if that point is deleted from the analysis.

Values above 1 indicate that a point is possibly influential. Cook's distances exceeding 4 indicate that the point has a major effect on the values of the parameter estimates.

Leverage. Leverage values identify *potentially* influential points. Observations with leverages two times greater than the expected leverages are potentially influential points.

The expected leverage of a data point is *p/n* where there are *p* parameters and *n* data points.

Because leverage is calculated using only the dependent variable, high leverage points tend to be at the extremes of the independent variables (large and small values), where small changes in the independent variables can have large effects on the predicted values of the dependent variable.

DFFITS. The $DFFITS_i$ statistic is a measure of the influence of a data point on regression prediction. It is the number of estimated standard errors the predicted value for a data point changes when the observed value is removed from the data set before computing the regression coefficients. Predicted values that change by more than 2.0 standard errors when the data point is removed are potentially influential.

Influence Diagnostics:

Row	Cook's Dist.	Leverage	DFFITS
1	0.1262	0.1265	-0.9006
2	0.0031	0.1147	0.1361
3	1.1749	0.1178	1.2982
4	0.0071	0.0922	-0.2053
5	0.0001	0.0834	-0.0184
6	0.0041	0.0735	0.1563
7	0.0223	0.0655	0.3683
8	1.2851	0.1534	1.4167
9	0.0301	0.0521	0.4326
10	0.0347	0.0468	0.4687

Figure 62: Regression Report Showing the Influence Diagnostics

Confidence and Prediction Intervals

The confidence level for both intervals has a default value of 95%. You can change it in the Report Options for Nonlinear Regression dialog box. More Statistics on page 86

Confidence intervals about the predicted values define a range of values where the population mean at the dependent variable is located with a certain probability. This probability is called the *confidence level*.

95% Confidence:

Row	Predicted	95% Conf-L	95% Conf-U	95% Pred-L	95% Pred-U
1	2.606061	0.930576	4.281545	0.140930	5.071191
2	2.833333	1.797303	3.869364	0.749352	4.917315
3	3.006494	1.904268	4.108719	0.888825	5.124162
4	3.140693	2.127688	4.153698	1.068062	5.213323
5	3.251082	2.337246	4.164919	1.225072	5.277092
6	3.352814	2.339809	4.365819	1.280183	5.425444
7	3.461039	2.358813	4.563265	1.343370	5.578707
8	3.590909	2.554878	4.626940	1.506928	5.674890
9	3.757576	2.082091	5.433060	1.292446	6.222706

Figure 63: The 95% Confidence Section of the Report

Row. This is the row number of the observation.

Predicted. This column shows the value for the dependent variable predicted by the regression model for each observation.

Confidence. The confidence interval for the regression gives the range of variable values computed for the region containing the true relationship between the dependent and independent variables, for the specified level of confidence. The 95% Conf-L values are lower limits and the 95% Conf-U values are the upper limits.

Prediction. The confidence interval for the population gives the range of variable values computed for the region containing the population from which the observations were drawn, for the specified level of confidence. The 95% Pred-L values are lower limits and the 95% Pred-U values are the upper limits.

Regression Equation Libraries and Notebooks

Regression equations are stored in notebook files just as other SigmaPlot documents. Notebooks that are used to organize and contain only regression equations are referred to as libraries, and distinguished from ordinary notebooks with a file extension of .jfl. These library notebooks can be opened and modified like any other notebook file. You can also use ordinary SigmaPlot notebooks (.jnb) as equation libraries, as well as save any notebook as a .jfl file.

Regression equations within notebooks are indicated with a regression symbol icon that appears next to the equation name.

The equations that appear in the Regression Wizard are read from a default regression library. The way the equations are named and organized in the equations panel is by using the section name as the category name, and the entry name as the equation name.

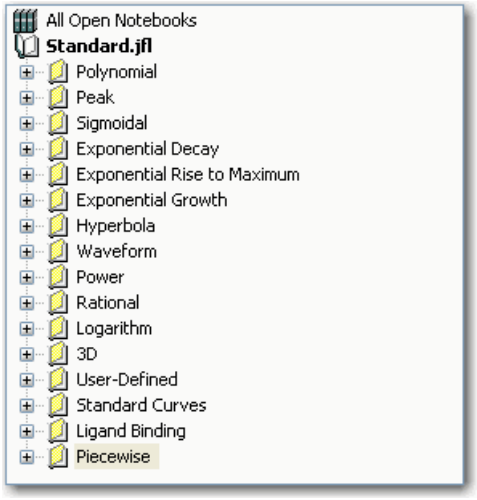


Figure 64: The Standard Regression Equation Library

For example, the **standard.jfl** regression library supplied with SigmaPlot has fourteen categories of built-in equations:

- Polynomial
- Peak
- Sigmoidal
- Exponential Decay
- Exponential Rise to Maximum
- Exponential Growth
- Hyperbola
- Waveform
- Power
- Rational
- Logarithm
- 3D
- Standard Curves
- Ligand Binding
- Piecewise
- Probability

These categories correspond to the section names within the **Standard.jfl** notebook.

To see the library currently in use, click **Back** in the **Regression Wizard** equation panel. Previously selected libraries and open notebooks can be selected from the **Library** drop-down list.

Opening an Equation Library

You can open, view, and modify a regression equation library as you would any ordinary notebook. To open a regression library:

1. Click the **Main Button** and then click **Open**, select *.jfl as the file type from the **File Type** drop-down list, then select the library to open, or
2. Click the **Open** button in the Regression Wizard library panel to open the current library. You can reach the library panel by clicking Back on the Equations panel.

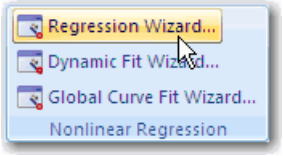
   You can copy, paste, rename and delete regression equations as any other notebook item. Opening a regression equation directly from a notebook automatically launches the Regression Wizard with the variables panel selected.

Using a Different Library for the Regression Wizard

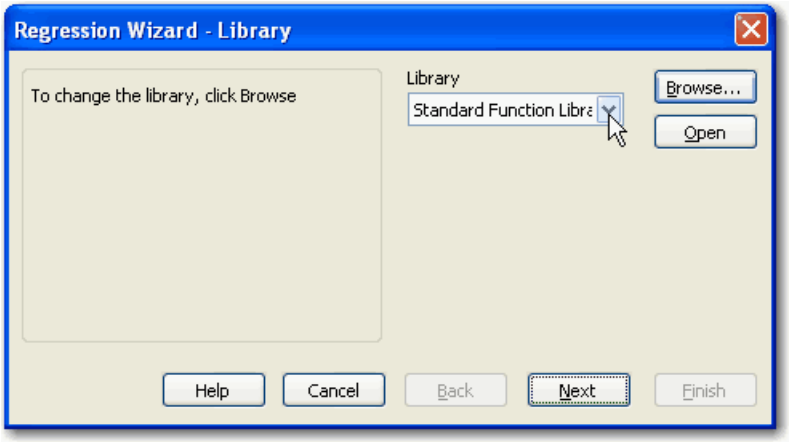
You can also select another notebook or library as the source for the equations in the Regression Wizard. Selecting a different equation library changes the categories and equations listed in the Regression Wizard equations panel.

To change the library:

1. Start the **Regression Wizard** by pressing **F5** or on the **Analysis** tab, in the **Nonlinear Regression group**, click **Regression Wizard**.



2. Click **Back** to view the library panel.



3. **To change the library,** enter the new library path and name, or click **Browse**.
4. In the **File Open** dialog box, change the path and select the file to use as your regression library. When you start the Regression Wizard next, it will continue to use the equation library selected in the library panel.

Note: Opening a regression equation directly from a notebook does not reset the equation library.

Curve Fitting Date And Time Data

You can run the Regression wizard on data plotted versus calendar times and dates. Dates within and near the twentieth century are stored internally as very large numbers. However, you can convert these dates to relatively small numbers by setting **Day Zero** to the first date of your data, then converting the date data to numbers. After curve fitting the data, you can switch the numbers back to dates. Setting Day Zero

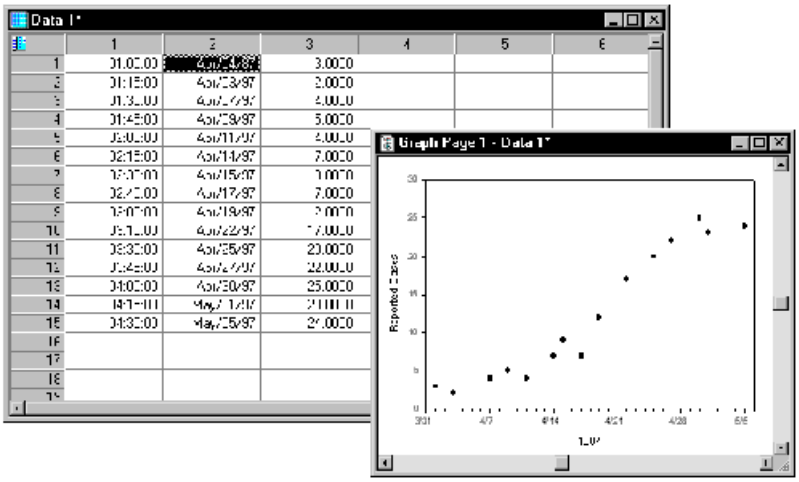
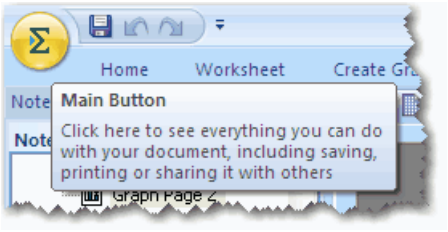


Figure 65: You can curve fit dates, but you must convert the dates to numbers first. Time only data (as shown in column 1) does not require a conversion.

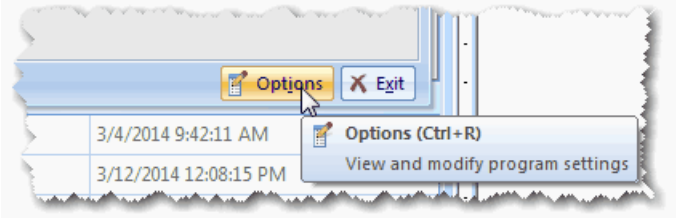
Tip: If you have entered clock times only, then you can directly curve fit those time without having to convert these to numbers. Time only entries assume the internal start date of 4713 B.C. (the start of the Julian calendar); however, if you have entered times using a more recent calendar date, you must convert these times to numbers as well.

Converting Dates to Numbers

1. Click the **Main Button**.

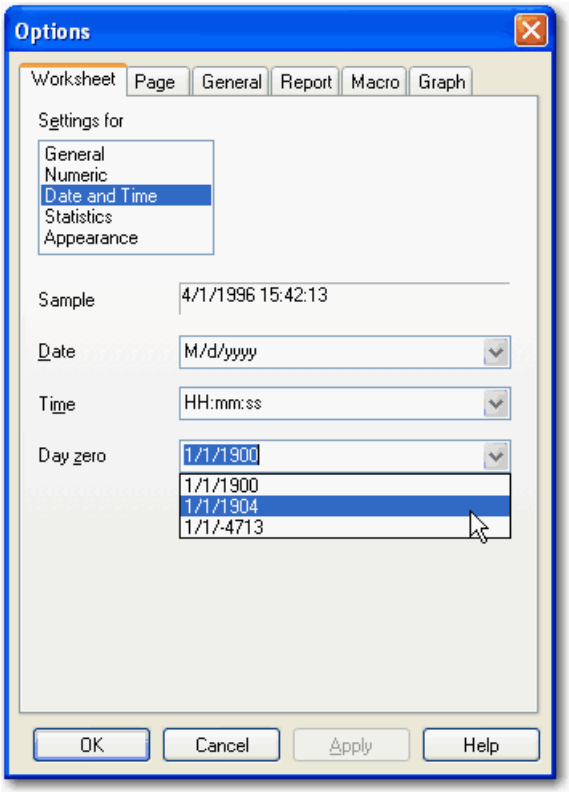


2. Click **Options**.



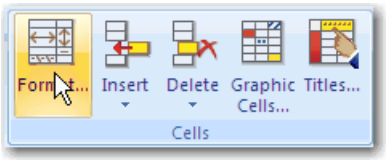
   The **Options** dialog box appears.

3. In the **Options** dialog box, click the **Worksheet** tab.
4. Select **Date and Time** from the **Settings for** list.

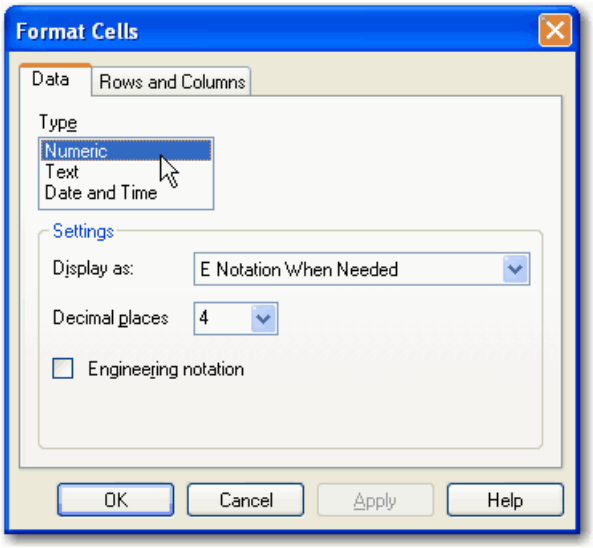


5. Set **Day Zero** to be the first date of your data, or to begin very close to the starting date of your data. You must include the year as well as month and day.
6. Click **OK**, then view the worksheet and select your data column.

7. On the **Worksheet** tab, in the **Cells** group, click **Format Cells**.



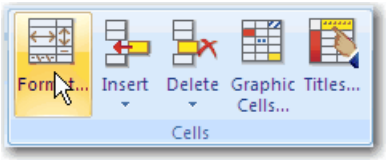
8. In the **Format Cells** dialog box, under **Type**, click **Numeric**.



9. Click **OK**.
   Your dates are converted to numbers. These numbers should be relatively small numbers. If the numbers are large, you did not select a Day Zero near your data starting date.
10. If the axis range of you graph is manual, convert it back to automatic. Select the axis, then using the **Property Browser**, change the range to **Automatic**. Changing Axis Range
11. Click you curve and run your regression. When you are finished, you must convert the original and fitted curve x variable columns back to dates.

Converting Numbers Back to Dates

1. Select each column.
2. On the **Worksheet** tab, in the **Cells** group, click **Format Cells**.



   The **Format Cells** dialog box appears.
3. In the **Format Cells** dialog box, under **Type**, click **Numeric**.
4. In the **Format Cells** dialog box, click the **Data** tab.
5. Under **Types**, select **Date and Time**.

6. On the **Date** drop-down list, click a date format.

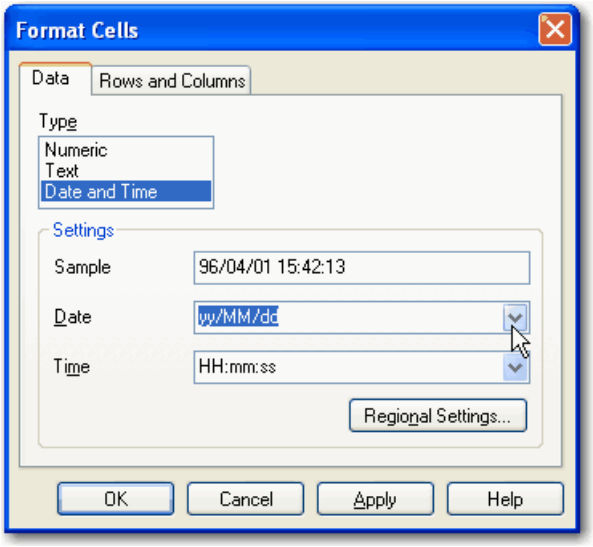


Figure 66: Converting Numeric Data back to Date and Time Data

7. Click **OK**.

When the columns are converted back to dates, the graph re-scales and you have completed your date and time curve fit.

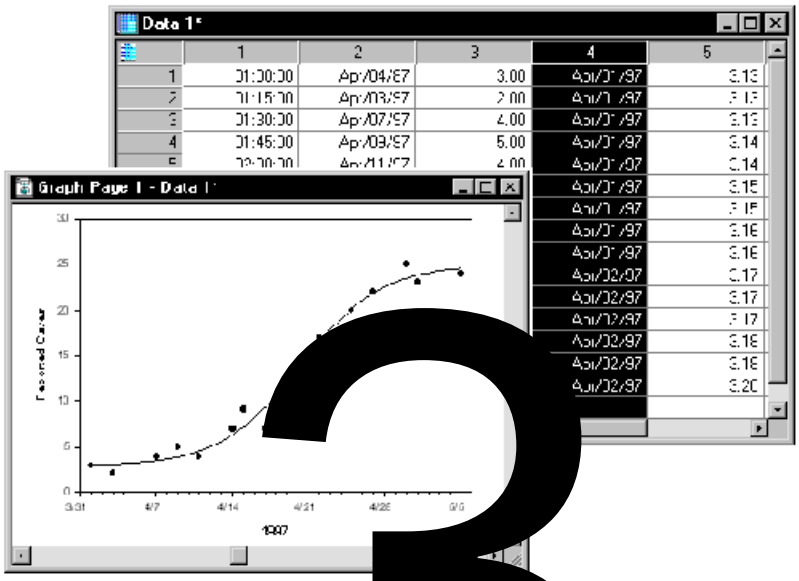


Figure 67: The Data and Fitted Curve X [illegible]les Converted Back to Dates and Graphed

Regression Results [illegible]sages

When the initial results of a reg[illegible] message about the completion status appears. Explanations of the[illegible] different messages are foun[illegible]

Converged, zero parameter changes. The changes in all parameters between the last two iterations are less than the computer's precision.

Did not converge, exceeded maximum number of iterations. More iterations were required to satisfy the convergence criteria. Select More Iterations to continue for the same number of iterations or increase the number of iterations specified in the Options dialog box and rerun the regression.

Did not converge, inner loop failure. There are two nested iterative loops in the Marquardt algorithm. This diagnostic occurs after 50 sequential iterations in the inner loop. The use of constraints may cause this to happen due to a lack of convergence. In some cases, the parameter values obtained with constraints are still valid, in the sense that they result in good estimates of the regression parameters.

Terminated by user. You pressed Esc, or selected the Cancel button and terminated the regression process.

Function overflow using initial parameter values. The regression iteration process could not get started since the first function evaluation resulted in a math error. For example, if you used f = sqrt(-a*x), and the initial a value and all x values are positive, a math error occurs. Examine your equation, parameter values and independent variable values, and make the appropriate changes.

Parameters may not be valid. Array ill conditioned on final iteration. During the regression iteration process the inverse of an array (the product of the transpose of the Jacobian matrix with itself) is required. Sometimes this array is nearly singular (has a nearly zero determinant) for which very poor parameter estimates would be obtained.

SigmaPlot uses an estimate of the "condition" of the array (ill conditioned means nearly singular) to generate this message (see Dongarra, J.J., Bunch, J.R., Moler, C.B., and Stewart, G.W., *Linpack User's Guide*, SIAM, Philadelphia, 1979 for the computation of condition numbers).

Usually this message should be taken seriously, as something is usually very wrong. For example, if an exponential underflow has occurred for all x values, part of the equation is essentially eliminated. SigmaPlot still tries to estimate the parameters associated with this phantom part of the equation, which can result in invalid parameter estimates.

A minority of the time the "correct," though poorly conditioned, parameters are obtained. This situation may occur, for example, when fitting polynomial or other linear equations.

Parameters may not be valid. Array numerically singular on final iteration. This is the limiting case of the above condition where the array cannot be inverted and the condition number is infinite. In this case, the parameter values are not well specified and their standard errors cannont be properly interpreted.

Parameters may not be valid. Overflow in partial derivatives. The partial derivatives of the function to be fit, with respect to the parameters, are computed numerically using first order differences.

Math errors from various sources can cause errors in this computation. For example if your model contains exponentials and the parameters and independent variable values cause exponential underflows, then the numerical computation of the partial derivative will be independent of the parameter(s). SigmaPlot checks for this independence.

Check the parameter values in the results screen, the range of the independent variable(s) and your equation to determine the problem.

There may be inconsistent constraints. Check constraint equations. This occurs if you have defined constraints like a>0 and a<-1.

Error Status Messages

Bad constraint. The regression cannot proceed because a constraint you defined either was not linear or contained syntax errors.

Invalid or missing 'fit to' statement. The regression lacks a fit to statement, or the fit to statement contains one or more syntax errors.

No observations to fit. The regression cannot proceed unless at least one x,y data pair (observation) is included. Check to be sure that the data columns referenced in the regression specifications contain data.

No parameters to fit. The regression specifications do not include any parameter definitions. To add parameter definitions, return to the Equation Options dialog box and type the parameter definitions in the Parameters edit window.

No weight statement. The regression specifications include a fit to statement with an unknown weight variable. Check the Variables edit window to see if a weight variable has been defined and that this corresponds to the variable in the regression statement.

Not enough or bad number of observations. In regression, the x and y data sets must be of the same size. The data sets (x and y columns) you specified contain unequal numbers of values.

Problem loading the file [Filename]. File too long; truncated. The fit file you tried to load is too long. Regression files can be up to 50 characters wide and 80 lines long. Any additional characters or lines were truncated when the file was loaded into the Edit Window.

Section has already been submitted. This regression section has already been defined.

Symbol [Variable or Function] has not been defined. The fit to statement in the regression definition contains an observed variable which is undefined, or the fit to statement in the regression definition contains an undefined function. Examine the regression specifications you have defined and be sure that the dependent variable listed in the regression statement exists and corresponds to the variable defined in the Variables edit window and that the function listed in the regression statement exists and corresponds to the function you defined in the Equations edit window.

Unreferenced variable. The regression specifications define a parameter that is not referenced in any other statements. Either delete the parameter definition, or reference it in another statement.

Chapter 4

Dynamic Curve Fitting

Topics:

Nonlinear curve fitting is an iterative process that may converge to find a best possible solution. It begins with a guess at the parameters, checks to see how well the equation fits, then continues to make better guesses until the differences between the residual sum of squares no longer decreases significantly. Curve-fitting Algorithm on page 80

For complicated curve fitting problems, use SigmaPlot's new Dynamic Fit Wizard to find the best the solution. The Dynamic Fit Wizard automates the search for initial parameter values that lead to convergence to the best possible solution. For example, typically for a user-defined function, you would need to edit the code to manually enter the initial parameter values, possibly repeatedly, until you find the best fit. Entering Initial Parameters on page 134

The Dynamic Fit Wizard takes away the guesswork of estimating the initial parameters. You enter a range for the minimum and maximum initial parameter values, and the Dynamic Fit Wizard does the rest, giving you the confidence that you'll find the best fit.

Using the Dynamic Fit Wizard

Like the Regression Wizard, the Dynamic Fit Wizard is a step-by-step guide through the curve fitting procedures, but with an additional panel in which you set the search options.

Note that the Dynamic Fit Wizard is especially useful for more difficult curve fitting problems with three or more parameters and possibly a large amount of variability in the data points. For linear regressions or less difficult problems, such as simple exponential two parameter fits, the Dynamic Fit Wizard is overkill and you should use the Regression Wizard.

Selecting the Data Source

1. View the page or worksheet with the data you want to fit.

   If you select a graph, right-click the curve you want fitted, and on the shortcut menu, click **Dynamic Fit Curve**.

   Remember: If you are running a regression from the graph page, make sure you select the plot itself, not the graph, or **Dynamic Fit Curve** will not appear on the shortcut menu.

   If you are using a worksheet, select the variables in the worksheet you want to fit, then:
2. On the **Analysis** tab, in the **Nonlinear Regression** group, click **Dynamic Fit Wizard**.

Selecting the Equation to Use

Select an equation from the **Equation Category** and **Equation Name** drop-down lists. You can view different equations by selecting different categories and names. The equation's mathematical expression and shape appear to the left. Regression Equation Libraries and Notebooks on page 107

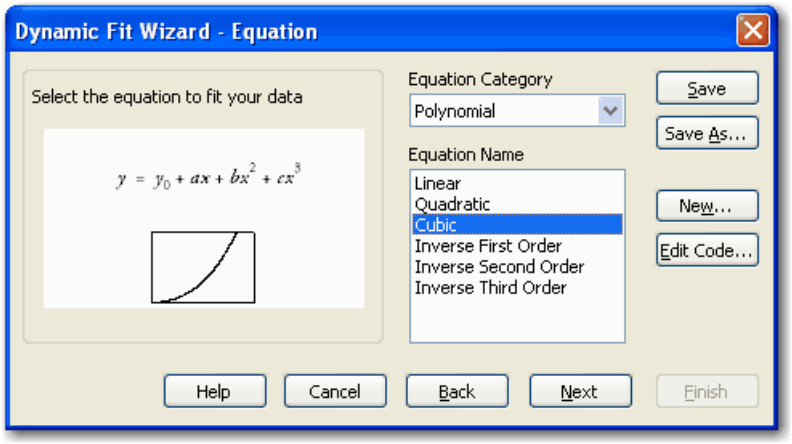


Figure 68: Selecting an Equation Category and Equation Name

If the equation you want to use isn't on this list, you can create a new equation. You can also browse other notebooks and regression equation libraries for other equations.

Note: SigmaPlot remembers the equation for the next time you open the Dynamic Fit Wizard.

If the **Finish** button is available, click it to complete your regression. If it is not available, or if you want to further specify your results, click **Next**.

Selecting the Variables to Fit

1. Click **Next** to open the variables panel. From here, you can select or re-select your variables. There are three ways to select variables:
   - Selecting a curve on a graph.
   - Selecting a column in a worksheet.
   - Selecting the variable from the **Variable Columns** drop-down list in the Dynamic Fit Wizard. The equation picture to the left prompts you for which variable to select.

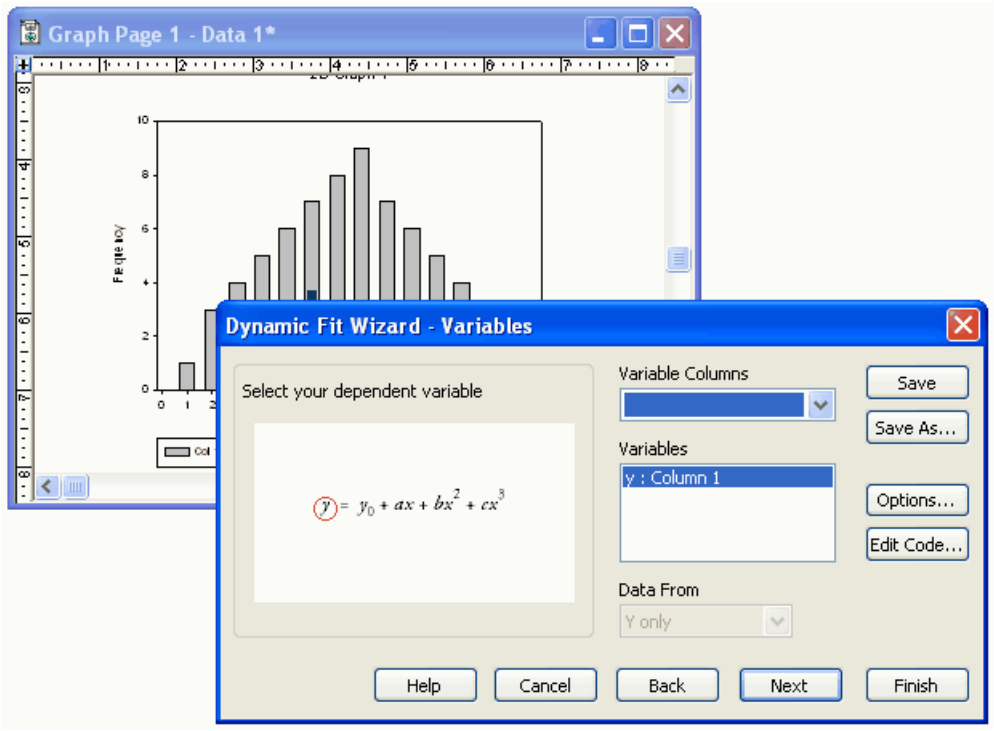


Figure 69: Selecting a plot as the data source for the Dynamic Fit Wizard.

You can also modify other equation settings and options from this panel by clicking **Options**, which opens the **Equations Options** dialog box. These options include changing initial parameter estimates, parameter constraints, weighting, and other related settings. Equation Options

If you pick variables from a worksheet column, you can also set the data format. Variable Options on page 90

2. When you have selected your variables, you can either click **Finish**, or click Next to view the Search Options. For more information, seeDynamic Fit Wizard - Search Options.

Setting the Dynamic Curve Fit Options

Unlike in the Regression Wizard, in the Dynamic Fit Wizard you can set the minimum and maximum ranges to search for initial parameter values. These are the values where, if you were to do this manually for a user-defined function, you would click Edit Code on the Variables panel of the Dynamic Fit Wizard (or Regression Wizard). The parameters appear under Initial Parameters.

Using the Regression Wizard, you would have to repeatedly enter the values until you found the best fit. Here, the Dynamic Fit Wizard does this for you. It selects a sequence of parameter estimates that are maximally distant from one another and in this way attempts to span the parameter ranges specified.

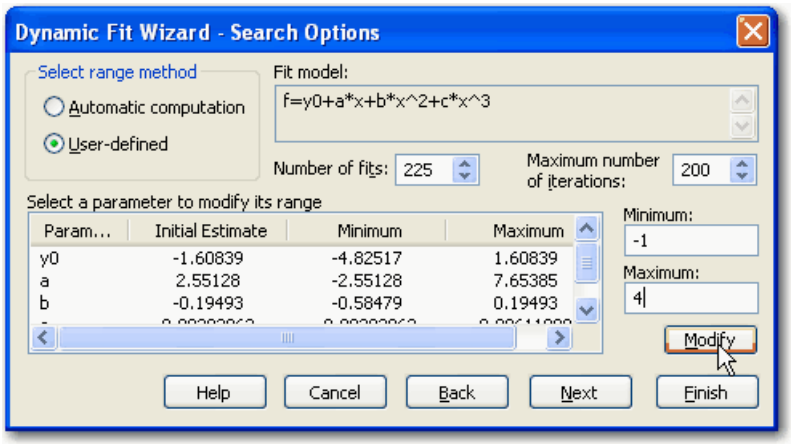


Figure 70: Setting the Search Options in the Dynamic Fit Wizard

The values under the **Parameter**, **Initial Estimate**, **Minimum** and **Maximum** columns contain the range information and the initial parameter estimates for each parameter in the equation file.

1. To set the number of fits and maximum number of iterations:
   a) Enter or select a number from the **Number of fits** drop-down list. The default value is 200. This is a good value to start with through for more difficult problems, you may want to increase it.
   b) Set the Marquadt-Levenberg algorithm from the **Maximum number of iterations** drop-down list. Again, more difficult problems may require a larger value. Check the “Iterations exceeding” percentage in the Dynamic Fit report. If this is greater than 50% then increase the Maximum number of iterations.

      Once the process exceeds this limit for a fit, then there is "no convergence" for this fit. The process continues with the next set of starting parameter values.
2. To change a parameter range:
   a) Under **Select range method** select **User-defined**.
   b) Select a parameter listing under **Select a parameter to modify its range**.
   c) Enter the ranges into the **Minimum** and **Maximum** boxes.
   d) Click **Modify**.

      These new values appear under the **Minimum** and **Maximum** columns.
   e) Click **Next**. As the dynamic curve fit procedure begins, a progress bar appears in the status bar in the lower left corner of SigmaPlot, indicating the proportion of fits as they are analyzed over time. Once the set **Number of Fits** has been reached, the **Dynamic Fit Wizard - Fit Results** panel appears. For more information, seeViewing Fit Results on page 118.

Viewing Fit Results

The fit results also appear if you receive a warning or error message about your fit.

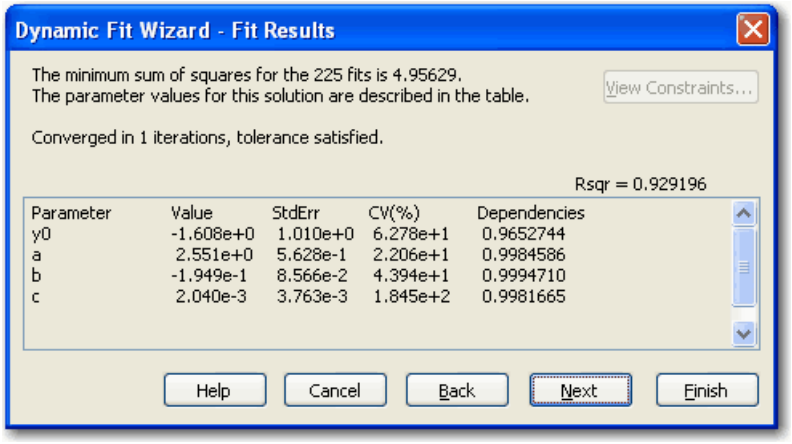


Figure 71: The Dynamic Fit Results for a Regression

If you wish to modify the remainder of the results that are automatically saved, click Next. Otherwise, click **Finish**. For more information, see Setting the Dynamic Curve Fit Options on page 117.

The subsequent panels provide options for the output data.

Chapter 5

Global Curve Fitting

Topics:

Use the Global Curve Fit Wizard when you want to fit an equation to several data sets simultaneously. The selected equation must have exactly one independent variable. The data sets can be selected from a worksheet or a graph using a variety of data formats. You can also specify the behavior of each equation parameter with respect to the data sets. A parameter can be localized to have a separate value for each data set, or a parameter can be shared to have the same value for all data sets.

For example, suppose your global curve fitting problem is based upon the three-parameter equation family y=a*x^2+b*x+c. If you select three data sets, then there will be at most nine different parameters, three for each data set. Suppose you decide to share parameter a, but allow parameters b and c to vary independently among the data sets. With one shared parameter and two local parameters, the total number of parameters drops to seven. The model for this problem takes the form:

$$y = \begin{cases} a1*x^2 + b1*x + c1 & \text{if}(x,y)\text{is in data set 1} \\ a1*x^2 + b2*x + c2 & \text{if}(x,y)\text{is in data set 2} \\ a1*x^2 + b3*x + c3 & \text{if}(x,y)\text{is in data set 3} \end{cases}$$

The Global Curve Fit Wizard is very similar to the Regression and Dynamic Fit Wizards in design and operation. The main difference is the extra panel for determining the shared parameters.

Using the Global Fit Wizard

Like the Regression and Dynamic Fit Wizards, the Global Fit Wizard provides a step-by-step guide through the curve fitting procedures, but with an additional panel in which you set shared parameters for multiple data sets with one equation model.

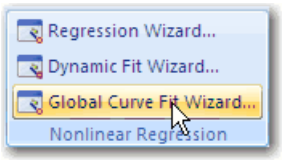
Selecting the Data Source

1. View the page or worksheet with the data you want to fit.

   If you select a graph, right-click the curve you want fitted, and on the shortcut menu, click **Global Fit Curve**.

   Remember: If you are running a regression from the graph page, make sure you select the plot itself, not the graph, or **Global Fit Curve** will not appear on the shortcut menu.

   If you are using a worksheet, select the variables in the worksheet you want to fit, then:
2. On the **Analysis** tab, in the **Nonlinear Regression** group, click **Global Curve Fit Wizard**.



3. In the **Global Fit Wizard**, click **Next**. For more information, see Selecting the Equation to Use on page 122.

Selecting the Equation to Use

Select an equation from the **Equation Category** and **Equation Name** drop-down lists. You can view different equations by selecting different categories and names. The equation's mathematical expression and shape appear to the left.

Remember: When selecting an equation, bear in mind that the Global Curve Fit Wizard only supports equations with one independent variable.

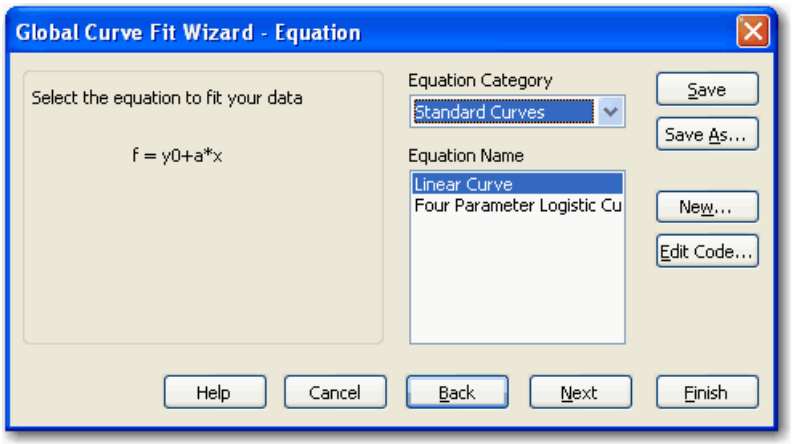


Figure 72: Selecting an Equation Category and Equation Name

If the equation you want to use isn't on this list, you can create a new equation. You can also browse other notebooks and regression equation libraries for other equations.

Tip: SigmaPlot remembers the equation for the next time you open the Global Fit Wizard.

If the **Finish** button is available, click it to complete your regression. If it is not available, or if you want to further specify your results, click **Next** to open the **Shared Parameters** panel.

Selecting the Parameters to Share

In **Shared Parameters** panel, you select the parameter variables that you'd like to share across the data sets. The equation you selected in the previous panel appears to the left.

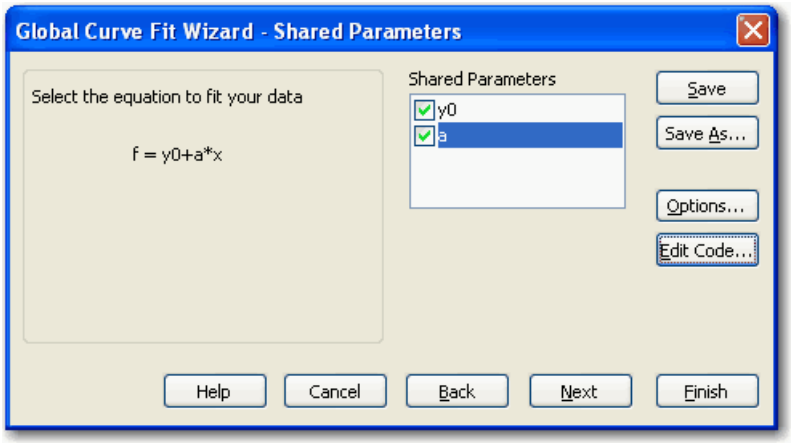


Figure 73: Select the parameters you'd like to share from the Shared Parameters list.

1. Modify other equation settings and options from this panel by clicking **Options**, which opens the **Equations Options** dialog box. These options include changing initial parameter estimates, parameter constraints, weighting, and other related settings.
2. Click **Edit Code** if you want to view and edit the fit text.

3. When you have selected the parameters you want to share, you can either click **Finish**, or click Next to select the variables to fit .

Selecting the Variables to Fit

In the **Variables** panel, you can select or re-select your the data for your dependent and independent variables. There are three ways to select variables:

v c

Viewing Fit Results

The fit results also appear if you receive a warning or error message about your fit.

Figure 75: The Global Fit Results for a Regression

If you wish to modify the remainder of the results that are automatically saved, click **Next**. Otherwise, click **Finish**. The subsequent panels provide options for the output data.

Setting Numeric Output Options

The **Numeric Output Options** panel lists:

- Which results are saved to the worksheet.
- Whether or not to generate a detailed statistical report of the regression.
- Whether or not a copy of the regression equation is saved to the section to the notebook that contains the data that was fitted.

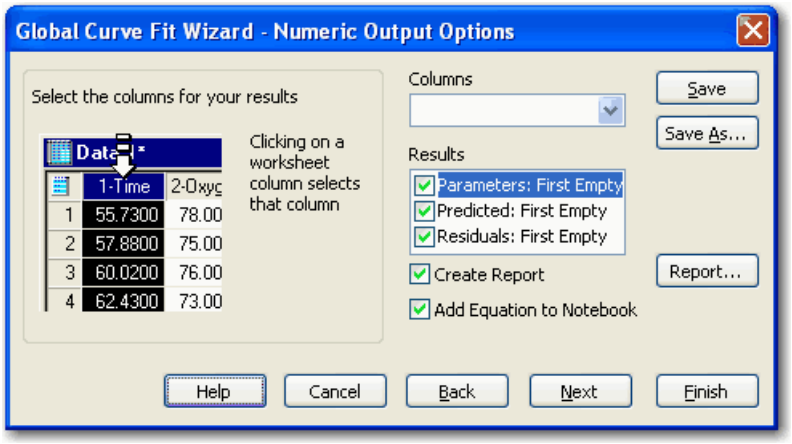


Figure 76: Selecting the results to save.

1. Select which results you want to keep from the **Results** list. These settings are remembered between regression sessions.
2. To set the options for the report, click **Report**.
3. Click **Next** to set the graph options.

Worksheet Result Options

This dialog box appears by clicking **More worksheet results** on the Global Fit Wizard - Numeric Output Options panel. You can select several types of calculations to compare and study fit results obtained from the Global Fit

Wizard. These results of these calculations appear in the worksheet for every (convergent) solution. Dynamic Fit Wizard - Numeric Output Options

Basic results - sum of squares, iteration counts, final parameters. Selected by default. Select this option to create one column for sum of squares, titled "SumSq", one column for the number of iterations needed for convergence, titled "Iterations", and *p* columns of the final, or best-fit, parameter values, where *p* is the number of parameters in the fit model. The column title for this is "<parameter value name>-Final".

All worksheet results are linked to the values in the sum of squares column. Its values are ordered from smallest to largest. Each row of data refers to a particular fit.

Clearing this option disables the Additional options.

Additional options. Select to place any of the following options below into the worksheet. To better distinguish columns of data, single blank columns are inserted into the worksheet between the groups of data specified by the options below.

- **Condition number.** Select to create a worksheet column called "Condition Number". This is the condition number of the covariance matrix obtained at the final iteration. For a positive-definite matrix, the condition number is computed as the ratio of the maximum and minimum eigenvalues. It is a measure of the sensitivity of the sum of squares value to a change in parameter values at the final iteration. Larger condition numbers indicate more uncertainty in specifying the best-fit parameters. The values in this column use E notation, with three decimal points of precision. Engineering and E Notation
- **Starting parameters.** These are the initial parameter estimates used to start the fit algorithm. These values are selected from the parameter ranges that you've specified. The title for this column is "<parameter-name>-Start".
- **Parameter standard errors.** These are the asymptotic standard errors for the parameters computed at the final iteration. They measure the range of uncertainty in specifying the best-fit parameters. The title of the Parameter standard errors columns is "<parameter name>-StdErr".
- **Coefficients of variation.** For each parameter value in the best-fit solution, this is the percent value of the ratio of the parameter's standard error to the parameter's absolute value. The title for this column is "<parameter name>-CV%".
- **Dependencies.** Given a parameter value in the best-fit solution, this is a number between 0 and 1 that measures the dependency of the rate of change of the predicted values with respect to the parameter on the rates of changes with respect to the other parameters. A value close to 1 indicates uncertainty in specifying the given parameter value in the best-fit solution. The title for this column is "<parameter name>-Dep".

Setting Graph Options

1. If you selected your variables from a graph, select **Add curve to** to automatically add the equation curve to that graph. This option on appears if you ran the regression using a graph curve as a data source.

   You can also plot the equation on any other graph on that page by selecting one from the drop-down list.
2. After selecting the graphed results you want, click **Finish**.

3. Select **Create new graph** to create a new graph of the original data and fitted curve.

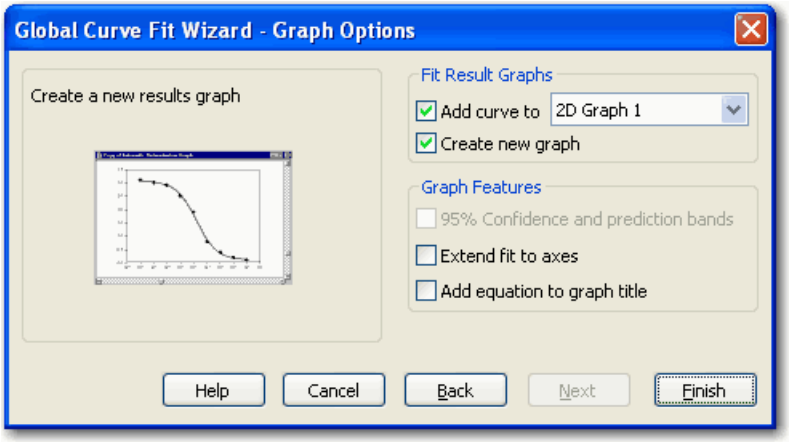


Figure 77: Selecting the results to graph. These settings are retained between sessions.

4. After selecting the graphed results you want, click **Finish**.
5. Select **Extend fit to axes** to extend the equation curve to intersect the Y-axes.

6. After selecting the graphed results you want, click **Finish**.
7. Select **Add equation to graph title** to insert the equation of the curve fit under the title of the graph.

8. After selecting the graphed results you want, click **Finish**.
9. After selecting the graphed results you want, click **Finish**.

Finishing the Global Fit

After clicking **Finish**, all your results are displayed in the worksheet, report, and graph. The initial defaults are to save parameter and computed dependent variable values to the worksheet, to create a statistical report, and to graph the results.

Global Curve Fit Reports

To create a Global Curve Fit report, make sure you select **Create Report** on the **Numeric Output Options** panel of the **Global Fit Wizard**.

When you click **Finish** in the **Global Fit Wizard**, a report appears. The report content depends on the report options that you have selected. The following sections are displayed when all options are selected:

Data Source and Equation Used. The notebook and worksheet names containing the data are displayed followed by the equation name and its functional form.

Data Set Specifications. The independent and dependent variable worksheet columns for each data set are tabulated.

Global Parameters. The equation parameters that have been selected to be shared (if any) are listed.

Global Goodness of Fit. Four goodness of fit statistics for the overall (global) fit are listed: R, R^2, *Adjusted* R^2 and the standard error of the estimate.

Analysis of Variance. Two ANOVA tables are displayed, without and with correction for the mean of the observations.

Statistical Tests. The results of the PRESS, Durbin-Watson, Normality, Constant Variance and (retrospective) Power statistics are displayed.

Individual Data Set Fit Results. This consists of the number of observations in each data set, two goodness of fit criteria (R^2 and residual sum of squares), the parameter estimates and their statistics (*std err*, *t*, *P* and *VIF*.

Individual Data Set Regression Diagnostics. Predicted values, raw residuals, standardized residuals, Studentized residuals, Studentized deleted residuals are displayed if selected.

Individual Data Set Influence Diagnostics. Cooks distance, leverage, and DFFITS are displayed.

Individual Data Set Confidence Limits. Upper and lower confidence and prediction limits are listed. The percentage confidence is typically 95% but any value between 1 and 99 may be used.

Fit Equation Description. The global curve fit algorithm generates a set of equations, one for each data set, that are used to fit to the multiple data sets. These equations, their initial parameter estimates, and other options are displayed here.

Chapter 6

Editing Code

Topics:

You can edit a regression equation by clicking the **Edit Code** button in the **Regression Wizard**, the **Dynamic Fit Wizard**, or the **Global Curve Fit Wizard**. This opens the Function dialog box. Regression equations can be selected from within the wizard, or opened from a notebook directly. You can also create new regression equations. Creating a new equation requires entry of all the code necessary to perform a regression.

About Regression Equations

Equations contain not only the regression model function, but other information needed by SigmaPlot to run a regression. All regression equations contain code defining the equations, parameter settings, variables, constraints, and other options used. To edit the code for an equation, you need to either open and edit an existing equation, or create a new equation.

Protected Code for Built-in Equations

All built-in equations provided in the standard equation library (**standard.jfl**) have protected portions of code which can be viewed and copied but not edited. However, you may use Add As to create a duplicate entry that can be edited, and you can also copy a built-in equation from the library to another notebook or section and edit it.

Opening an Existing Equation

You can open an equation by:

- Double-clicking an equation icon in the notebook window, or selecting the equation then clicking **Open**.
- Starting either the **Regression Wizard**, **Dynamic Fit Wizard** or the **Global Curve Fit Wizard**, then selecting the equation by category and name.

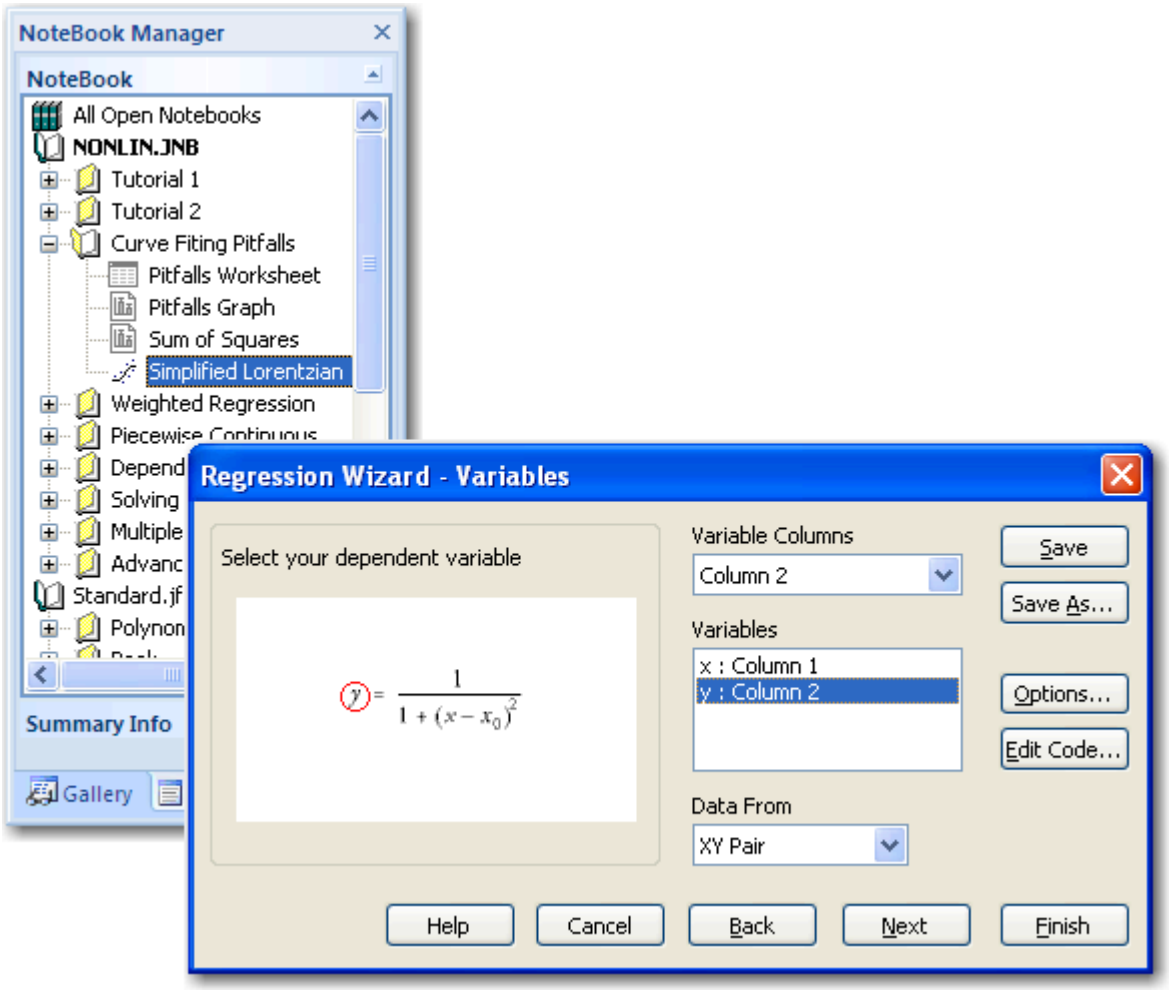


Figure 78: Opening an equation from the Notebook Manager

You can also double-click an equation in a notebook while any one of these three wizards is open to switch to that equation. Once an equation is opened, you can edit it by clicking the **Edit Code** button.

Creating a New Equation

If you require an equation that does not appear in the standard equation library (standard.jfl), you can create a new equation using the Functions dialog box

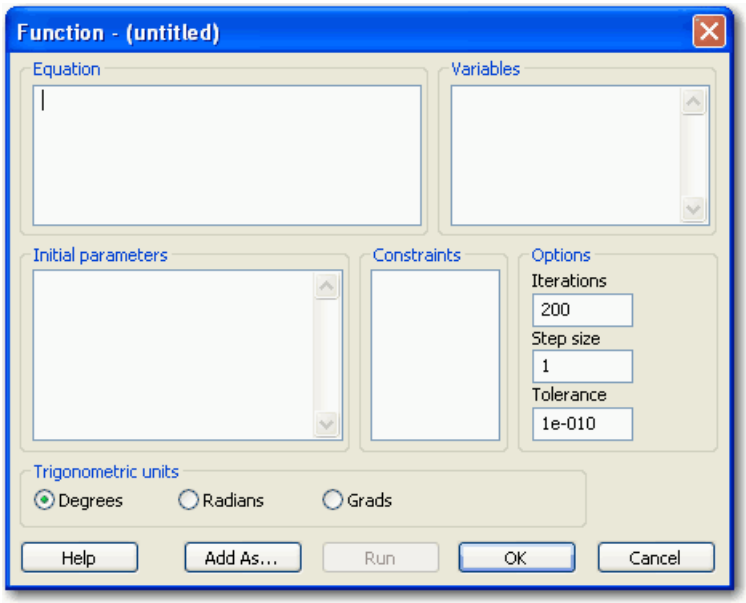
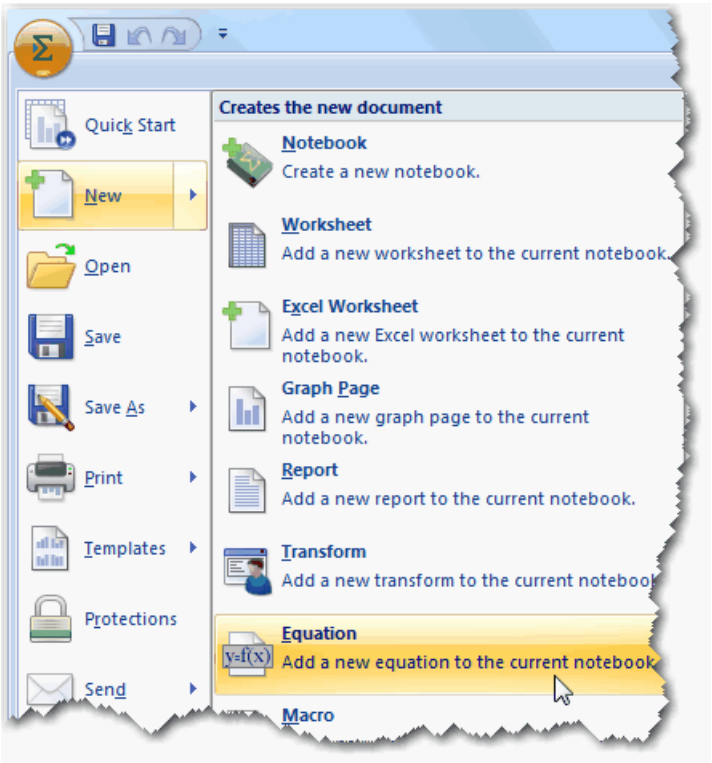


Figure 79: You can create equations of your own in the Function dialog box.

You can open the Functions dialog box by:

- Clicking the **New** button in the **Regression Wizard**, the **Dynamic Fit Wizard**, or the **Global Curve Fit Wizard**.
- Clicking **Main Button** > **New** > **Equation**.



- Right-clicking in the **Notebook Manager** and choosing **New Equation** from the shortcut menu. A new equation document has no default settings for the equations, parameters, variables, constraints, or other options.

To create a new equation from within a Wizard:

1. On the **Analysis** tab, in the **Nonlinear Regression** group, click **Nonlinear Regression**, **Dynamic Fit Wizard**, or **Global Curve Fit Wizard**.
2. Click the **New** button view the **Function** dialog box and then create new equation.
3. To create an equation from the Notebook Manager, right-click the section where you want the new equation to be added. If you want the equation to be created in a new section, right-click the notebook icon.
4. On the shortcut menu, click **New** from the shortcut menu, and then click Equation. The **Function** dialog box appears.

Copying Equations

You can copy an existing equation from any notebook view to another, and modify it as desired.

Adding Equations as New Entries

To edit equations from within the **Regression Wizard**, the **Dynamic Fit Wizard**, or the **Global Curve Fit Wizard**, and add them as new equations to the current library, click the **Add As** button in the **Function** dialog box. The **Add As** dialog box appears, where you can enter a name for the new equation name.

Entering Regression Equation Code

To enter the code for new equations, click the desired edit window in the Function dialog box and enter your code.

Tip: Open the Function dialog box by clicking **New** or **Edit Code** on the **Regression Wizard**, the **Dynamic Fit Wizard**, or the **Global Curve Fit Wizard**.

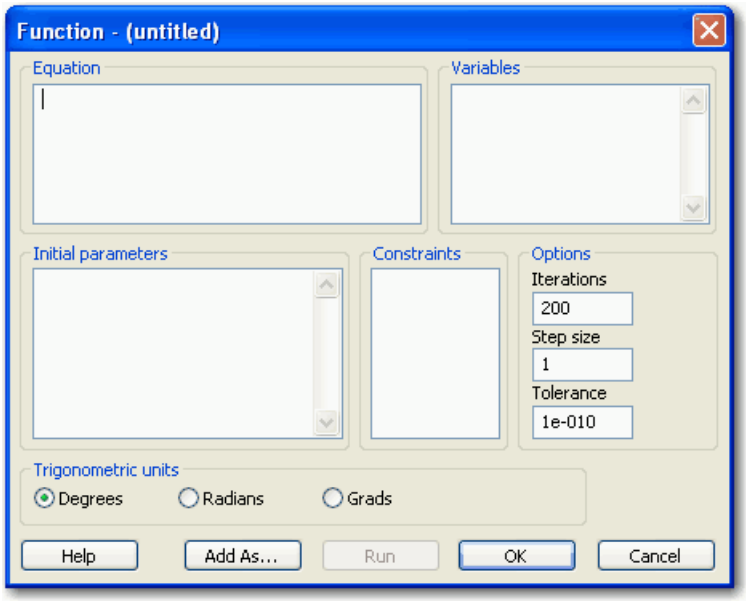


Figure 80: The Function dialog box

This section covers the minimum steps required to enter the code for a regression equation. For more information on entering the code for each section, see:

- Equations.
- Variables.
- Weight Variables.
- Initial Parameters.
- Constraints.

- Other Options.

Adding Comments

Place comments in the edit box by preceding them with an apostrophe ('), or a semicolon (;). You can also use apostrophes or semicolons to comment out equations instead of deleting them.

Entering Equations

To enter the code for the Equation section:

1. Click in the Equation window and type the regression equation model, using the transform language operators and functions.

   The equation should contain all of the variables you plan to use as independent variables, as well as the name for the predicted dependent variable (which is not your y variable). You can use any valid variable name for your equation variables and parameters, but short, single letter names are recommended for the sake of simplicity. Omit the observed dependent variable name from the regression model. The observed dependent variable (typically your y variable) is used in the fit statement.
2. Press the Enter key when finished with the regression equation model, then type the fit statement. The simplest form of the fit statement is: fit f to y Where f is the predicted dependent variable from the regression model, and y is the variable that will be defined as the observed dependent variable (typically the variable plotted as y in the worksheet).

   You can also define whether or not weighting is used.

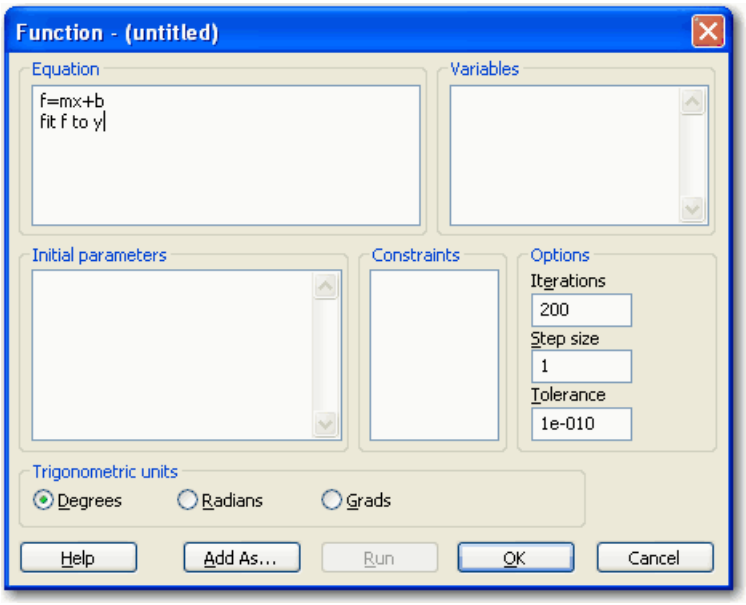


Figure 81: Entering the regression equation and the regression statement.

Example: The code f=m*x+b fit f to y can be used as the model for the function , and also defines y as the observed dependent variable. In this example, x is the independent variable, and m and b the equation parameters.

Defining Constants

Constants that appear in the equations can also be defined under the Equation section. If you decide to treat an equation parameter as a constant rather than as a regression parameter, you can define the value for that constant here, then make sure you don't enter this value in the Initial parameters section. The other way to make a parameter a constant is to enter an equality constraint for that parameter in the Constraints section.

Constants defined here appear under the Constants option in theEquation Options dialog box.

Entering Variables

Independent, dependent, and weighting variables are defined in the Variables section. One of the variables defined must be the observed values of the dependent variable: that is, the "unknown" variable to be solved for. The rest are the independent variables (predictor, or known variables) and an optional weighting variable.

To define your variables:

1. Click in the Variables section and type the character or string you used for the first variable in your regression equation.
2. Type an equal sign (=), then enter a range for the variable. Ranges can be any transform language functions that produce a range, but typically are worksheet columns.

   Note: The variable values used by the Regression Wizard, the Dynamic Fit Wizard, or the Global Curve Fit Wizard depend entirely on what are selected from the graph or worksheet; the values entered here are only used if the From Code data format is selected, or if the regression is run directly from the Function dialog box.
3. Repeat these steps for each variable in your equation. Up to ten independent variables can be defined, but you must define at least one variable for a regression equation to function. The curve fitter checks the variable definitions for errors and for consistency with the regression equation.

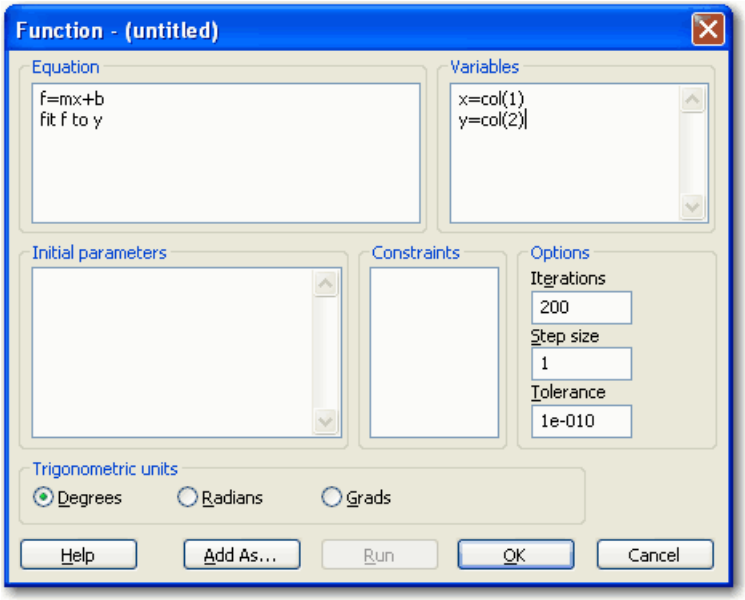


Figure 82: Entering the variable definitions

Example: To define x and y as the variables for the equation code f=m*x+b, fit f to y you could enter the code x=col(1), y=col(2) which defines an x variable as column 1 and a y variable as column 2, using these columns whenever the regression is run directly from the code.

Automatic Initial Parameter Estimation Functions

Any user-defined functions you plan on using to compute initial parameter estimates must be entered into the Variables section.

Entering Initial Parameters

Parameters are the equation variables and offset constants that you are trying to estimate in your equation model. The definitions or functions entered into the Initial Parameters sections determine which variables are used as parameters in your equation model, and also their initial values for the curve fitter.

The curve fitter checks the initial parameter values for errors and for consistency with the regression equations.

To enter initial parameter values:

1. Click in the Initial parameters section and type the name of the first parameter as it appears in your equation model, followed by an equals (=) sign.
2. Enter the initial parameter value used by the curve fitter. Ideally, this should be as close to the real value as possible. This value can be numeric, or a function that computes a good guess for the parameter.

   Using a function for the initial parameter value is called automatic parameter estimation.

   Example: If your data for the equation code f=m*x+b, fit f to y appear to rise to the right and run through the origin, you could define your initial parameters to be m=0.5, b=0. These are good initial guesses, since the m coefficient is the slope and the b constant is the y-intercept of a straight line.

Parameter Constraints

Parameter Constraints are completely optional. Use them to limit the parameter ranges to meaningful values for your particular problem. For more information, see .

Options

The Iterations, Step Size and Tolerance options sometimes can be used to improve or limit your curve fit. The default settings work for the large majority of cases, so you do not need to change these setting unless truly required.

Saving Equations

Once you are satisfied with the settings you have entered into the Function dialog box, you can save the equation. Clicking OK automatically updates the equation entry in the current notebook or regression library. If you created a new equation, you are prompted to name it before it is added to your notebook. If you are editing an existing equation, you can click Add As to add the code as a new equation to the current library or notebook. In order to save your changes to disk, you must also save the notebook or library. You can save changes before you close the wizard by clicking Save. Click Save As to save the regression library to a new file. If your equation is part of a visible notebook, you can save changes by saving the notebook using the Save button or Save or Save As on the Main Button.

Note that when an equation is edited using the Equation Options dialog box, all the changes are also automatically updated and saved.

Saving Equation Copies with Results

You can save equations along with the targeted page or worksheet while saving your regression results. Just select the Add Equation to Notebook option on the Numeric Output Optionspanel, and a copy of the equation used is added to the same section as reports and other results.

Equations

The Equation section of the Function dialog box defines the model used to perform the regression as well as the names of the variables and parameters used. The regression equation code is defined using the transform language operators and functions. The equation must contain all of the variables you wish to use. These include all independent variables and the predicted dependent variable. All parameters and constants used are also defined here. The Equation code consists of two required components:

- The *equation model* describing the function(s) to be fit to the data.
- The *fit statement*, which defines the predicted dependent variable and, optionally, the name of a weighting variable.

Any constants that are used must also be defined under the Equation section.

Defining the Equation Model

The equation model sets the predicted variable (called f in all built-in equations) to be a function of one or more independent variables (called x in the built-in two-dimensional Cartesian equations) and various unknown coefficients, called parameters.

The model may be described by more than one function. For example, the following three equations define a dependent variable f, which is a constant for x < 1 and a straight line for x ≥ 1.

```
f = if (x < 1, constant (x), line (x))
constant (x) = c
line (x) = a + b * x
```

Number of Parameters

You can enter and define up to 500 parameters, but a large number of parameters will slow down the regression process. You can determine if you are using too many parameters by examining the parameter dependencies of your regression results. Dependencies near 1.0 (0.999 for example) indicate that the equation is overparameterized, and that you can probably remove one or more dependent parameters. For more information, seeInterpreting Fit Results on page 95.

Defining the Fit Statement

The most general form of the fit statement is:

```
fit f to y with weight w
```

f identifies the predicted dependent variable to be fit to the data in the set of equations, as defined by the model.

y is the observed dependent variable, later defined in the Variables section, whose value is generally determined from a worksheet column.

w is the optional weight variable, also defined in the Variables section.

Any valid variable name can be used in place of *f*, *y*, and *w*.

If the optional weighting variable is not used, the fit statement has the form:

```
fit f to y
```

Defining Constants

Define constants by setting one of the parameters of the equation model to a value, using the form

```
constant=value
```

For example, one commonly used constant is pi, defined as *pi*=3.14159265359

Defining Alternate Fit Statements

You can create alternate fit statements that call different weight variables. These statements appear as fit statements preceeded by two single quotes (', not a double quote).

For each weight variable you define, you can create a weighting option by adding commented fit statements to the equation window.

For example, an Equation window that reads:

```
f=a*exp(-b*x)+c*exp(-d*x)+g*exp(-h*x)
```

```
fit f to y with Weight Reciprocal
```

```
''fit f to y
```

with weight Reciprocal causes the option Reciprocal to be displayed in the Fit with weightlist on the Equation Optionsdialog box.

Variables

Independent, dependent, and weighting variables are defined in the Variables edit window of the Function dialog box. One of the variables defined must be the observed values of the dependent variable: that is, the observed variable to be fitted. The rest are the independent variables (predictor, or known variables) and any optional weighting variables. Up to ten independent variables can be defined.

To define your variables, select the Variables edit window, then type the variable definitions. You generally need to define at least two variables—one for the dependent variable data, and at least one for the independent variable data.

Variable Definitions

Variable definitions use the form:

```
variable = range
```

You can use any valid variable name, but short, single letter names are recommended for the sake of simplicity (for example, *x* and *y*). The range can either be the column number for the data associated with each variable, or a manually entered range.

Most typically, the range is data read from a worksheet. The curve fitter uses SigmaPlot's transform language, so the notation for a column of data is:

```
col(column,top,bottom)
```

The *column* argument determines the column number or title. To use a column title for the column argument, enclose the column title in quotation marks. The *top* and *bottom* arguments specify the first and last row numbers and can be omitted. The default row numbers are 1 and the end of the column, respectively. If both are omitted, the entire column is used. For example, to define the variable x to be column 1, enter:

```
x = col(1)
```

Data may also be entered directly in the variables section. For example, you can define y and z variables by entering:

```
y = {1,2,4,8,16,32,64}
```

```
z = data(1,100)
```

This method can have some advantages. For example, in the example above the data function was used to automatically generate *z* values of 1 through 100, which is simpler than typing the numbers into the worksheet.

Note: The **Regression Wizard**, **Dynamic Fit Wizard**, and **Global Curve Fit Wizard** generally ignores the default variable settings, although they require valid variable definitions in order to evaluate an equation. Variables are redefined when the variables are selected from a worksheet or graph. However, you can force

the use of the hard-coded variable definitions, either by selecting **From Code** as the data source, or running the regression directly from the Function dialog box.

Transform Language Operations

You can use any transform language operator or function when defining a variable. For example:

```
x = 10^data(-2, log(10.8),0.5)
```

```
y = ((col(2)-col(2)*(.277*col(1))^0.8))*1.0e-12
```

```
z = 1/sqrt(abs(col(3)))
```

are all valid variable names.

User-Defined Functions

Any user-defined functions that are used later in the regression code can be defined in either the **Variables** section or the **Equations** section.

Concatenating Columns

Constructor notation can be used to concatenate data sets. For example, you may want to fit an equation simultaneously to multiple y columns paired with one x column. If the x data is in column 1 and the y data is in columns 2 through 6, you can enter the following variable statements:

```
x = {col (1), col (1), col (1), col (1), col (1)}
```

```
y = {col (2), col (3), col (4), col (5), col (6)}
```

The variable x is then column 1 concatenated with itself four times, and variable y is the concatenation of columns 2 through 6.

If the function to be fit is f, then the fit statement

```
fit f to y
```

fits *f* to the dependent variable values in columns 2 through 6 for the independent variable values in column 1.

Weight Variables

Variables used to perform weighted regressions are known as weight variables. All weight variables must be defined along with other variables in the Variables section.

Specifying the Weight Variable to Use

The use of weighting is specified by code in the Equation section, which can call weight variables defined in the Variables section. Weight variables are selected from the fit statement, using the syntax:

```
 fit f to y with weight w
```

where *w* is the weight variable defined in the Variables section. A weight variable is often defined in terms of a worksheet column where each row value is used to compute the weight that is assigned to the observation in the same row of the dependent variable column. In particular, it is common to define the weight variable as the reciprocal of either the observed dependent variable or its square. For example, if *y*=col(2) is the observed dependent variable, the weighting variable can defined as 1/col(2) or as 1/col(2)^2. For more information, see Equations on page 135.

For many applications, the weights have fixed values that do not change during the regression process. However, you can define a weight variable to be a function of the parameters. More specifically, the weight variable can be defined in terms of the predicted values or the regression residuals. This type of weighting is often used when the observation errors are not known in advance.

For example, in “weighting by predicteds”, the weight variable can be defined as w = 1/f, where f defines the predicted values of the fit model. This type of weighting is useful when the errors are Poisson distributed. For this distribution, the population means, which the predicted values are estimating, are equal to the population variances, which is why the weight variable is defined as the reciprocal of the predicted values.

Another way to define a weight variable is in terms of the regression residuals. For example, w = 1/(1 +4*(y-f)^2) defines the weights in terms of the residuals y-f, where y are the observed values of the dependent variable and f are the predicted values computed from the regression model. Such weight functions are used in “robust regression” to mitigate the effects of outliers on the regression results by assigning small weights to observations with large residuals.

Defining Optional Weight Variables

You can define more than one possible weight variable, and select the one to use from the **Equation Options** dialog box. Simply create multiple weight variables, then create alternate fit statement entries selecting the different weight variables in the Equations section.

When to Use Weighting

Least squares regression problems usually assume that the errors at all data points are equal. When the error variance is not homogeneous, weighting should be used. If variability increases with the magnitude of the dependent variable value, larger dependent variable values will have larger residuals. Large residuals will cause the squared residuals for large dependent variable values to overwhelm the smaller residuals. The total sum of squares will be sensitive only to the large dependent variable values, leading to an incorrect regression if weighting is not used.

You may also weight the regression when there is a requirement for the curve to pass through some point. For example, the (0,0) data point can be heavily weighted relative to the other data points to force the curve through the origin.

Note: When using weight variables, the calculated parameters are valid, but the interpretation of their statistics depends on which option is used for computing their standard errors. Statistical Summary Table

The Weighting Process: Norm and Residuals Changes

The weight values are proportional to the reciprocals of the variances of the dependent variable. Weighting multiplies the corresponding squared term in the sum of squares, dividing the absolute value of the residual by its standard error. This causes all terms of the sum of squares to have a similar contribution, resulting in an improved regression.

For weighted least squares, the weights w are included in the sum of squares to be minimized.

$$\sum_{i=1}^{n} w_i \left(y_i - \hat{y}_i\right)^2$$

When weighting is used, the norm that is computed and displayed in the **Progress** dialog box is $\sqrt{SS}$ and includes the effect of weighting. The residuals computed are the weighted residuals.

Initial Parameters

The code under the Initial parameters section of the Function dialog box specifies which equation coefficients and constants to vary and also sets the initial parameter values for the regression. To enter parameters, select the Initial parameters section, then type the parameters definitions using the form:

```
parameter=initial value
```

All parameters must appear in the equation model. All equation unknowns not defined as variables or constants must be defined in Initial Parameters.

Initial Parameter Values

For the initial values, a "best guess" may speed up the regression process. If your equation is relatively simple (only two or three parameters), the initial parameter values may not be important. For more complex equations, however, good initial parameter values can be critical for a successful convergence to a solution.

Automatic Parameter Estimation

All built-in equations use a technique called *automatic parameter estimation*, which computes an approximation of the function parameters by analyzing the raw data. You can indicate the parameter value you wish to appear as the Automatic setting by typing two single quotes followed by the string Auto after the parameter setting. For example, entering the parameter line

```
a=max(y) ''Auto
```

tells the Equation Options dialog box to use

```
max(y)
```

as the Automatic parameter value for *a*. Automatic Determination of Initial Parameters on page 140

Automatic Determination of Initial Parameters

SigmaPlot automatically obtains estimates of the initial parameter values for all built-in equations found in the standard equation library (standard.jfl). When automatic parameter estimation is used, you no longer have to enter static values for parameters yourself—the parameters determine their own values by analyzing the data.

Tip: It is only important that the initial parameter values are robust among varying data sets, i.e., that in most cases the curve fitter converges to the correct solution. The estimated parameters only have to be a "best guess" (somewhere in the same ballpark as the real values, but not right next to them). You can create your own methods of parameter determination using the new transform function provided just for this purpose.

The general procedure is to smooth the data, if required, and then use functions specific to each equation to obtain the initial parameter estimates.

Consider the logistic function as an example. This function has the stretched *s* shape that changes gradually from a low value to a high value or vice versa. The three parameters for this function determine the high value (*a*), the x value at which the function is 50% of range of the function's amplitude (x0) and the width of the transition (*b*). As expressed in the transform language, the function is entered into the Equation window as f=a/(1+exp(-(*x*-x_0\)/*b*)) fit *f* to *y*. Noise in the data can lead to significant errors in the estimates of x_0 and *b*. Therefore, a smoothing algorithm is used to reduce the noise in the data and three functions are then used on the smoothed data to obtain the parameter estimates.

To estimate the parameter *a* the maximum use the *y* value. Use the *x* value at 50% of the amplitude to estimate x_0, and the difference between the *x* values at 75% and 25% of the amplitude is used to estimate *b*. As entered into the Initial Parameters window, these are:

```
a=max(y)
''Auto b=xwtr(x,y,.5)/4
''Auto x0=x50(x,y,.5)
''Auto
```

Both the fwhm and xwtr transform functions have been specifically designed to aid the estimation of function parameters. Example 1: Curve Fitting Pitfalls on page 152

The "Auto comment that follows each parameter is used to identify that parameter value as the Automatic setting from within the Equation Options dialog box. Note that these values may not at all reflect the final values, but they are approximate enough to prevent the curve fitter from finding false or invalid results.

Alternate Parameter Values

You can insert alternate parameter values that appear in the Equation Options dialog box Initial Parameter Values drop-down lists. To add an alternate, insert a new line after the default value, then type two single quotes, followed by the alternate parameter setting. For example, the two lines `d=-F(0)[2] ''Auto, ''d=0.01` cause an alternate value of 0.01 to appear in the Equation Options dialog box Initial Parameter Values drop-down list for *d*. Alternate parameter values are automatically inserted when different parameter values are entered into the Equation Options dialog box.

Constraints

Linear parameter constraints are defined under the Constraints section. A maximum of 25 constraints can be entered. Use of constraints is optional. Constraints are used to set limits and conditions for parameter values, restricting the regression search range and improving regression speed and accuracy. Liberal use of constraints in problems which have a relatively large number of parameters is a convenient way to guide the regression and avoid searching in unrealistic regions of parameter space.

Valid Constraints

A constraint must be a linear function of the parameters using an equality (=) or inequality (< or >). For example, the following constraints for the parameters a, b, c, d, and e are valid:

```
a<1
10*b+c/20 > 2
d-e = 15
a>b+c+d+e
```

whereas

```
a*x<1
```

is illegal since *x* is not a constant, and

```
b+c^2>4
d*e=1
```

are illegal because they are nonlinear.

Tip: Although the curve fitter checks the constraints for consistency, you should still examine your constraint definitions before executing the regression. For example, the two constraints: `a<1 a>2` are inconsistent. The parameter *a* cannot be both less than 1 and greater than 2. If you execute a regression with inconsistent constraints, a message appears in the Results dialog box warning you to check your constraint equations.

Other Options

You can use several special options to influence regression operation. The different options can be used to speed up or improve the regression process, but their use is optional. The three options are:

- *Iterations*, the maximum number of repeated regression attempts. Iterations on page 142
- *Step Size*, the limit of the initial change in parameter values used by the regression as it tries different parameter values. Step Size on page 142
- *Tolerance*, one of the conditions that must be met to end the regression process. Tolerance on page 142

When the absolute value of the difference between the norm of the residuals from one iteration to the next is less than the tolerance, this condition is satisfied and the regression considered to be complete. Options are entered in the Options section edit boxes. The default values are displayed for new equations. These settings will work for most cases, but can be changed to overcome any problems encountered with the regression, or to perform other tasks, such as evaluating parameter estimation.

Iterations

Setting the number of iterations, or the maximum number of repeated regression attempts, is useful if you do not want to regression to proceed beyond a certain number of iterations, or if the regression exceeds the default number of iterations.

The default iteration value is 200. To change the number of iterations, simply enter the maximum number of iterations in the Iterations edit box.

Evaluating Parameter Values Using 0 Iterations

Iterations must be non-negative. However, setting Iterations to 0 causes no iterations occur; instead, the regression evaluates the function at all values of the independent variables using the parameter values entered under the Initial Parameters section and returns the results.

If you are trying to evaluate the effectiveness of automatic parameter estimation function, setting Iterations to 0 allows you to view what initial parameter values were computed by your algorithms.

Using zero iterations can be very useful for evaluating the effect of changes in parameter values. For example, once you have determined the parameters using the regression, you can enter these values plus or minus a percentage, run the regression with zero iterations, then graph the function results to view the effect of the parameter changes.

Step Size

The initial step size used by the Marquardt-Levenberg algorithm is controlled by the Step Size option. The value of the Step Size option is only indirectly related to changes in the parameters, so only relative changes to the step size value are important.

The default step size value is 1. To change the step size value, type a new value into the edit box. The step size number equals the largest step size allowed when changing parameter values. Changing the step size to a much smaller number can be used to prevent the curve fitter from taking large initial steps when searching around suspected minima. Example 1: Curve Fitting Pitfalls on page 152

If you are familiar with this algorithm, step size is the inverse of the Marquardt parameter.

Tolerance

The Tolerance option controls the conditions that must be met in order to end the regression process. When the absolute value of the relative change between the norm of the residuals from one iteration to the next is less than the tolerance, the regression process is considered to be complete.

The curve fitter uses two stopping criteria:

- When the absolute value of the difference between the norm of the residuals (square root of the sum of squares of the residuals), from one iteration to the next, is less than the tolerance value, the iteration stops.
- When all parameter values stop changing in all significant places, the regression stops.

When the tolerance condition has been met, a minimum has usually been found.

The default value for tolerance is 1.0e-10. To change the tolerance value, type the required value in the Tolerance edit box. The tolerance number sets the value that must be met to end the iterations.

More precise parameter values can be obtained by decreasing the tolerance value. If there is a sharp change in the sum of squares response surface near the minimum, then decreasing the tolerance from the default value will have little effect on the parameter values.

However, if the response surface is shallow about the minimum (indicating a large variability for one or more of the parameters), then decreasing tolerance can result in large changes to parameter values. It may also cause the regression process to run much longer. Example 1: Curve Fitting Pitfalls on page 152

Chapter 7

Regression Lessons

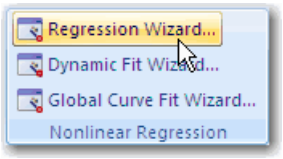
Topics:

This tutorial is designed to familiarize you with regression fundamentals. The sample graph and worksheet files for the tutorials are located in the Regression Examples (nonlin.jnb) notebook provided with SigmaPlot.

Lesson 1: Linear Curve Fit

In this lesson, you will fit a straight line to existing data points.

1. Open the **Tutorial 1 Graph** in the **Regression ExampleS** (nonlin.jnb)notebook and examine the graph. For more information, see About SigmaPlot's User and Program Files. The points appear to nearly follow a straight line.
2. On the **Analysis** tab, in the **Nonlinear Regression** group, click **Regression Wizard**.



   The **Regression Wizard** displays lists of equations by category. If the Linear equation is not already selected, select the **Polynomial** category and select **Linear** as the equation name.
3. Click **Next** to proceed. The next panel prompts you to pick your x, or independent variable.
4. Click the curve on the page to select it. Note that clicking the curve selects both the x and y variables for you.
5. Click **Next**. The Fit Results dialog box appears, displaying the progress of the fitting process. When the process is completed, the initial regression results are displayed.
6. Examine the results. The first result column is the parameter values; the intercept is -.94 and the slope is 1.24.

   The next column is an estimate of the standard error for each parameter. The intercept has a standard error of about 0.36 — not that good — and the slope has a standard error of about 0.10, which isn't bad.

   The third column is the coefficient of variation (CV%) for each parameter. This is defined as the standard error divided by the parameter value, expressed as a percentage. The CV% for the intercept is about 38.2%, which is large in comparison to the CV% for the slope (about 8.7%).

   The dependencies are shown in the last column. If these numbers are very close to 1.0, they indicate a dependency between two or more parameters, and you can probably remove one of them from your model.

Adding a Parameter Constraint

To make *y* always positive when *x* is positive, you cannot have a negative *y* intercept. You can recalculate the regression with this condition by constraining the parameter y_0 to be positive. That way *y* will never be negative when $x>0$.

1. On the Fit Results panel, click **Back**. The Variables panel appears.
2. Click **Edit Code**. The **Functions - Linear** dialog box is appears. Enter a value of `y0>0` into the **Parameter Constraints** edit box. This defines the constraint `y0>0`, which forces the *y* intercept to be positive.
3. Click **OK**, then click Next to refit the data with a straight line, this time subject to the constraint `y0>0`. When the fit results are displayed, the value for y_0 is now very close to zero, and the slope has slightly decreased to a value of approximately 0.98.
4. Click **View Constraints**; the **Constraints** dialog box appears with the constraint `y0>0` flagged with the label "(active)" indicating that the value of y0 is very close to the boundary value of the constraint which is zero.

   Note: Nonlinear regressions may find parameters that satisfy the constraints without having to activate some or all of the constraints. Constraints that are not used are not flagged as (active).
5. Click **OK** to return to the **Fit Results** panel, then click **Next** to proceed. For more information, see Saving Results on page 146.

Saving Results

You can select the results to save for a regression. These results are destroyed by default each time you run another regression equation.

You can save some of your results to a worksheet, and other results to a text report. To save worksheet results, make sure the results you want saved are checked in the **Results** list on the **Numeric Output Options**panel. You have the

option to save parameter values, predicted dependent (y) variable values for the original independent (x) variable values, and the residuals about the regression for each original dependent variable value.

1. To save a text report, select the **Create Report** option. The report for a nonlinear regression lists all the settings entered into the **Function** dialog box, a table of the values and statistics for the regression parameters, and some regression diagnostics.
2. You can also save a copy of the regression equation you used to the same notebook section as the page or worksheet on which you ran the regression. Select **Add Equation to Notebook** to save a copy of your equation.
3. Click **Next** to proceed. For more information, see .

Graphing Results

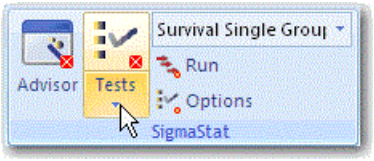
To plot the regression fit on the existing graph:

1. Select **Add curve** to Graph #1 on the **Graph Options** panel.
2. Click **Finish** to display your report and graphed results.

Comparing Regression Wizard Results with Linear Regression Results

The original data for this graph could have been fitted automatically in SigmaPlot with a linear regression using the **Linear** test. However, because you cannot specify constraints for the regression coefficients, a first order regression gives different results. To add a linear regression to your original data plot:

1. Select the plot of your original data by clicking it on the graph.
2. On the **Analysis** tab, in the **Statistics** group, click the **Tests** drop-down list.



3. Click **Regression** and then click **Linear**.
4. Select either the **Each Curve** or **All data in plot** option and pick a dotted line type for the regression line.
5. Click **OK** to accept the regression settings, then view the graph. Note the difference between the regression and the fitted line.

> **Tip:** If you had not used a parameter constraint, the result of the nonlinear regression would have been identical to the linear regression. Save the graph and worksheet to a file.

Lesson 2: Sigmoidal Function Fit

This tutorial leads you through the steps involved in a typical nonlinear regression for a "real world" scenario.

Examining and Analyzing the Data

The data used for this tutorial represents blood pressure measurements made in the neck (carotid sinus pressure), and near the outlet of the heart (the mean arterial pressure).

These pressures are inversely related. If the blood pressure in your neck goes down, your heart needs to pump harder to provide blood flow to your brain. Without this immediate compensation, you could pass out every time you stood up.

Sensors in your neck detect changes in blood pressure, sending feedback signals to the heart. For example, when you first get out of bed in the morning, your blood tends to drain down toward your legs. This decreases the blood pressure in your neck, so the sensors tell the heart to pump harder, preventing a decrease in blood flow to the brain.

You can do an interesting experiment to demonstrate this effect. Stand up and relax for a minute, then take your pulse rate. Count the number of pulses in 30 seconds, then lie down and immediately take your pulse rate again. Your pulse

rate will decrease as much as 25%. (Your heart doesn't have to pump as hard to get blood to the brain when you are lying down.)

1. Open the **Tutorial 2** graph file by double-clicking the graph page icon in the **Tutorial 2** section in the Regression Examples (nonlin.jnb) notebook.
2. Examine the graph. The two pressures are clearly inversely related. As one rises, the other decreases. The shape appears to be a reverse sigmoid, suggesting the use of a sigmoidal equation.

   A sigmoid shaped curve looks like an S that has had its upper right and lower left corners stretched. In this case, the S is backwards, since it starts at a large value, then decreases to a smaller value.

   Various forms of the sigmoid function are commonly used to describe sigmoids. In this case, you will use the four parameter sigmoid function provided in the standard regression library.
3. Right-click the curve and choose **Curve Fit**. The **Regression Wizard** appears.
4. Select **Sigmoidal** as your equation category, and **Sigmoid, 4 Parameter** as your equation.
5. Click **Next** twice. If you have correctly selected the curve, the **Iterations** dialog box appears, displaying the value for each parameter and the Sum of Squares for each iteration.

   Important: The iterations proceed more slowly than those for the linear fit. This is because the equation is much more complex and there are more parameters. Watch the Sum of Squares value decrease — this number is an index of the fit closeness, and decreases as the fit improves.

   When the fit condition is satisfied, the fit results are displayed.
6. Examine the results. The first column displays the parameter value, and the next column displays the estimated standard error. The third column is the coefficient of variation (CV%) for each parameter. (Note that these CV% values are unrealistically good — the largest is about 3.9%. Generally, CV% values for physiological measurements are greater than 5%.)

   True nonlinear regression problems (like this sigmoidal fit, but unlike a linear fit) have CV% values that are not absolutely correct. However, they still can be used to compare the relative variability of parameters. For example, b (3.9) is more than eight times as variable as c (0.45).

   None of the dependencies shown in the last column are close to 1.0, suggesting that the model is not over-parameterized.
7. To save the regression results and graph the curve, click **Finish**. A report along with worksheet data and a fitted curve are added to your notebook, worksheet and graph.

Fitting with a Different Equation

More than a single regression can be run and plotted on a graph. Typically, this is done to gauge the effects of changes to parameter values, or to compare the effect of a different fit equation.

In this case, try a five parameter logistic function instead of the four parameter version.

1. Press **F5**. The **Regression Wizard** appears. Select **Sigmoid, 5 Parameter** as your equation.
2. Click **Next**, then click **Options**. Enter a value of **5** into the **Iterations** box.
   The iterations option specifies the maximum number of iterations to perform before displaying the current results. You can see if a long regression is working correctly by limiting the number of iterations to perform. If the regression does not complete within the number of iterations specified, you can continue by clicking the **More Iterations** button in the **Fit Results** panel.
3. Compare the curve fits visually. As expected, the five parameter function appears to fit slightly better.
4. Click **OK**.
5. Click **Next** to calculate the new fit. The Fit Results panel appears and says, “Did not converge, exceeded maximum number of iterations.”
6. Note that the **More Iterations** option is enabled. Click **More Iterations** twice.
   The regression continues to completion, converging in 14 iterations.
7. Compare the curve fits visually. As expected, the five parameter function appears to fit slightly better.

8. Examine the results. The Sum of Squares value, standard deviations and CV% values are smaller than for the four parameter fit, indicating that this may be a better fit. However, two of the dependencies are close to 1.0, suggesting that the additional parameter may not have been needed.
9. Compare the curve fits visually. As expected, the five parameter function appears to fit slightly better.

Chapter 8

Advanced Regression Examples

Topics:

Example 1: Curve Fitting Pitfalls

This example demonstrates some of the problems that can be encountered during nonlinear regression fits.

Peaks in chromatograph data are sometimes fit with sums of Gaussian or Lorentzian distributions. A simplified form of the Lorentzian distribution is:

$$y = \frac{1}{1+(x-x_0)^2} \quad -\infty < x < \infty$$

where x_0 is the location of the peak value.

Open the Pitfalls worksheet and graph by double-clicking the Pitfalls Graph in the Regression Examples (nonlin.jnb) notebook. Note the positions of data points on the curve.

Open the Simplified Lorentzian regression equation by double-clicking it in the Regression Examples notebook. The Regression Wizard opens and displays the variables panel.

Click one of the symbols on the graph so that the selected Variables are Columns 1 and 2 .

The object is to determine the peak location x_0 for the data. Since this data was generated from the Lorentzian function above using $x_0 = 0$, the regression should always find the parameter value $x_0 = 0$.

How the Curve Fitter Finds the Peak Value Location

To find x_0, the curve fitter computes the sum of squares function:

$$\sum_{i=1}^{3}\left[f(x_i)-y_i\right]^2$$

as a function of the parameter x_0. The curve fitter then searches this parameter space for any x_0 value where a relative minimum exists.

The sum of squares for x_0 has two minima—an absolute minimum at $x_0 = 0$ and a relative minimum at $x_0 = 4.03$—and a maximum at 2.5. As the curve fitter searches for a minimum, it may stumble upon the local minimum and return an incorrect result. If you start exactly at a maximum, the curve fitter may also remain there.

False convergence caused by a small step size. Click **Options**. Note that the value of x_0 is set to 1000, and **Step Size** is set to 0.000001.

Click **OK**, then click **Next**.

Using the large initial value of x0 and a small step size, the curve fitter takes one small step, finds that there is no change in the sum of squares using the default value for tolerance (0.0001), and declares the tolerance condition is satisfied. The very low slope in the sum of squares at this large x_0 value causes the regression to stop.

False convergence caused by a large step size and tolerance. Click **Back**, then click **Options**. Open the **Step Size** list and select 100 .

Click **OK**, then click **Next**. The curve fitter takes a large step, reaches negative x_0 values, and finds a value $x_0 = -546$ for which the tolerance is satisfied.

The sum of squares function asymptotically approaches the same value for both large positive and negative values of x, so the difference of the sum of squares for $x_0 = 1000$ and $x_0 = -546$ is within the default value for the tolerance.

Reducing tolerance for a successful convergence. Click **Back**, then click **Options** again. Change the Tolerance value to 0.000001, then click **OK**.

Click **Next**. The regression continues beyond $x_0 = -546$ and successfully finds the absolute minimum at $x_0 = 0$.

Summary

When you used a poor initial parameter value, you needed to use a large initial step size to get the regression started, and you had to decrease the tolerance to keep the regression from stopping prematurely. Poor initial parameters can

arise also when using the Automatic method of determining initial parameters as well as when constant values are used.

You will now use initial parameter values which result in convergence to a local minimum and a local maximum.

Finding a local minimum. Click **Back**, then click **Options**.

Change the initial value of x_0 to 10 from the **Parameter Values** drop-down list.

In the **Tolerance** drop-down list, change the tolerance back to the default value of 0.0001, then click **OK**.

Click **Next**. The regression converges to $x_0 = 4.03$, which corresponds to the local minimum.

In this example, you know that a local minimum was found by viewing the sum of squares function for the single parameter x_0. However, when there are many parameters, it is usually not obvious whether an absolute minimum or a local minimum has been found.

Finding a local maximum. Click **Back**, then click **Options**. Change the initial parameter value of x_0 to 2.5, then click x_0.

Click **Next**. Because this initial parameter value happens to correspond to the maximum of the sum of squares function, the regression stops immediately. The slope is zero within the default tolerance, so the curve fitter falsely determines that a minimum has been found.

Finding the absolute minimum. Click **Back**, then click **Options**. Change the initial value of x_0 to 2.0.

Click **OK** to close the Options dialog box, then click **Next** to execute the regression. The initial parameter value is reasonably close to the optimum value, so the regression converges to the correct value $x_0 = 0.0$.

These last examples demonstrate how the curve fitter can find a local minimum and even a local maximum using poorly chosen initial parameter values.

Example 2: Weighted Regression

The data obtained from the lung washout of intravenously injected dissolved Xenon 133 is graphed in the Weighted Graph in the Weighted Regression section of the Regression Examples (nonlin.jnb) notebook.

Open the Weighted worksheet and graph by double-clicking the graph page icon in the Weighted section of the Regression Examples notebook.

The data in the graph displays the compartmental behavior of Xenon in the body. Three behaviors are seen: the wash-in from the blood (rapid rise), the washout from the lung (rapid decrease), and the recirculation of Xenon shunted past the lung (slow decrease).

The sum of three exponentials (a triple exponential) is used as a compartmental model:

$$CountRate = a_1 e^{-\lambda_1 t} + a_2 e^{-\lambda_2 t} + a_3 e^{-\lambda_3 t}$$

Least squares curve fitting assumes that the standard deviations of all data points are equal. However, the standard deviation for radioactive decay data increases with the count rate. Radioactive decay data is characterized by a Poisson random process, for which the mean and the variance are equal. Weighting must be used to account for the non-uniform variability in the data. These weights are the reciprocal of the variance of the data.

For a Poisson process, the variance equals the mean. You can use the inverse of the measurements as an estimate of the weights. The initial weighting variable only needs to be proportional to the inverse variance.

Double-click the Weighted Triple Exponential equation in the Weighted Regression section.

Click **Edit Code**, and examine the Variable value:

w = 1/col(2)

This sets w to equal the reciprocal of the data in column 2. Click **Cancel** to close the dialog box.

Click the data points to select your variables. To use the w variable as the weighting variable, click **Options**, and select *w* as the **Fit With Weight** value.

Click **OK** to close the dialog box.

Click **Next** to run the regression. The curve fitter finds a solution quickly.

Click **Finish** to complete the regression.

What would be the result without weighting? Double-click the Weighted Triple Exponential equation, then **Options**. Change the weighting to **(none)**, then click **OK**.

Click **Finish**. The curve fitter goes through the maximum number of iterations without converging.

Click **More Iterations** to continue as many times as needed for the fit to converge.

Click **Finish** again and view the Weighted graph page.

The graph shows the nonlinear regression results with and without weighting. The weighted results fit the very small recirculation data (represented by the third exponential) quite well. However, when weighting was not used, the curve fitter ignored the relatively small values in the recirculation portion of the data, resulting in a poor fit.

Example 3: Piecewise Continuous Function

The Piecewise section of SigmaPlot fit functions in standard.jfl contains piecewise linear functions. But these functions need not be linear. This is an example of a function that is a piecewise combination of a straight line and a one-minus-exponential function.

The piecewise line-exponential function consists of two functions - straight line and a one-minus-exponential - defined before and after the join-point x_0 .

$f_1 = a_1 + b_1 *x \; 0 < x <= x_0$

$f_2 = a_2 + b_2 \; (1 – exp(-c(x – x_0))) \; x_0 < x < infinity$

To make the piecewise function smooth and continuous, you need to satisfy the two conditions:

- The functions are equal at x_0

  $f_1 \; (x_0) – f_2 \; (x_0)$

- And the slopes of the functions are equal at x_0.

  $f_1 \; '(x_0) = f_2 \; '(x_0)$

The first condition gives

$a_1 + b_1 *x_0 = a_2$

while the second gives

$b_1 = b_2 *c$

Using these conditions you can rewrite the equations as

$f_1 = a_1 + b_1 *x \; 0 < x <= x_0$

$f_2 = (a_1 + b_1 *x_0) + (b_1 /c)*(1 - exp(-c*(x-x_0))) \; x_0 < x < infinity$

Double-click Piecewise Continuous Graph in the section Piecewise Continuous section of the Regression Examples (nonlin.jnb) notebook in the Samples directory of SigmaPlot's Program Files. The Piecewise Continuous graph page and worksheet appear. The data appears to be described by a straight line followed by a curve to a plateau.

View the notebook and double-click the Piecewise Continuous Regression equation.

The Regression Wizard - Variables panel appears.

View the graph page and select a data point on the graph (x:Column1, y:Column2), then click **Next** to run the regression. The equation with parameters a_1, b_1, x_0 and c is fit to the data.

Click **Finish** and view the graph page. A smooth transition at the join-point = 5.3 between the line and the one-minus-exponential is shown.

Example 4: Using Dependencies

This example demonstrates the use of dependencies to determine when the data has been "over-parameterized." Too many parameters result in dependencies very near 1.0. If a mathematical model contains too many parameters, a less complex model may be found that adequately describes the data. Sums of exponentials are commonly used to characterize the dynamic behavior of compartmental models. In this example you model data generated from the sum of two exponentials with one, two, and three exponential models, and you examine the parameter dependencies in each case.

Dependencies Over a Restricted Range

The first fit is made to data over a restricted range, which does not reveal the true nature of the data.

Open the Dependencies worksheet and graph by double-clicking the graph page icon in the Dependencies section of the Regression Examples (nonlin.jnb) notebook. The data generated from the sum of two exponentials:

$$f(t) = 0.9e^{-t} + 0.1e^{-0.2t}$$

is graphed on a semi-logarithmic scale over the range 0 to 3.

Although the data is slightly curved, the "break" associated with the two distinct exponentials is not obvious.

Right-click the curve and click **Fit Curve** on the shortcut menu to open the Regression Wizard .

Select the **Exponential Decay** category (click **Back** to change to the Standard Function Library, if necessary) and the **Single, 2 Parameter Exponential Decay** equation, then click **Next** twice.

The results show that the dependencies are not near 1.0, indicating that the single exponential parameters, a_1 and b_1, are not dependent on one another.

Click **Back** twice, and change the equation to the **Double, 4 Parameter Exponential** decay equation. Click **Next** twice.

The results show that the parameter dependencies for the double exponential are acceptable, indicating that they are unlikely to be dependent, and that using a double exponential produces a better fit (the curve fitter in fact finds the exact parameter values used to generate the data, producing a perfect fit with an R^2 of 1).

Dependencies Over an Extended Range

Click **Back** twice, and change the equation to a **Triple, 6 Parameter exponential decay** equation. Click **Next** twice.

The results show that the parameter dependencies for *a, b, c*, and *d* are 1.00, suggesting that the three exponential model is too complex and that one exponential may be eliminated. Click **Cancel** when finished.

Example 5: Solving Nonlinear Equations

You can use the nonlinear regression to solve nonlinear equations. For example, given a y value in a nonlinear equation, you can use the nonlinear regression to solve for the x value by making the x value an unknown parameter.

Consider the problem of finding the LD_{50} of a dose response experiment. The LD_{50} is the function of the four parameter logistic equation, which can be written as:

$$f(x) = \frac{a_1}{1+e^{b(x-c)}} + d$$

where *x* is the dose and *f(x)* is the response. You can use nonlinear regression to find the value for *x* which satisfies the equation:

$$50 = \frac{a_1}{1+e^{b(x-c)}} + d$$

Open the Solving Nonlinear Equation worksheet and graph file by double-clicking the graph page icon in the Solving Nonlinear Equations section of the Regression Examples (nonlin.jnb) notebook. Note that the value for *x* at *y* = 50 appears to be approximately 150.

Double-click the Solving Nonlinear Equation and click **Edit Code**.

Examine the regression statements. Note that *x* is a parameter, *y* = 0, and the fit equation is modified:

```
f = p1/(1 + exp(p2*(x -p3))) + p4 - 50
```

Since you are fitting *f* to = 0, these statements effectively solve the original problem for *x* when *y* = 50. The values for parameters *a*, *b*, *c*, and *d* were obtained by fitting the four parameter logistic equation to a given set of dose response data.

Click **Finish** to execute the regression. The parameter solution is the unknown *x*. For this example, *x* is approximately 149.5.

Example 6: Multiple Function Nonlinear Regression

You can use the Regression Wizard to fit more than one function at a time. This process involves combining your data into additional columns, then creating a third column which identifies the original data sets. In the most general case, the functional form of the equations used to fit each data set can differ. When all data sets are fit with equations of the same functional form, and parameters are either local to each data set or shared across all data sets (global), you can use the Global Curve Fitter. For more information, see Global Curve Fitting on page 121.

This example fits three separate equations to three data sets.

f1(x)=t(xE1)n1+(xE1)n

f2(x)=t(xE2)n1+(xE2)n

f3(x)=t(xE1)n1+(xE3)n

Open the Multiple Function worksheet and graph by double-clicking the graph page icon in the Multiple Function section of the Regression Examples (nonlin.jnb) notebook. The data points are for three dose responses.

Columns 1 and 2 hold the combined data for the three curves. Column 3 is used to identify the three different data sets. A 0 corresponds to the first dataset, 1 to the second, and 2 to the third.

Double-click the Multiple Functions Equation. The Regression Wizard opens with the variables panel displayed. Click Edit Code .

Examine the fit statements. The fit equation is an if statement which uses different equations depending on the value of d, which is the data set identifier variable. If *d* = 0, the data is fit to f1: if *d* = 1, the data is fit to f2; and if *d* = 2, the data is fit to f3.

The functions share the *T* and *n* parameters, but have individual *E* parameters of E_1, E_2, and E_3.

Click **Run** to execute the regression. The fit proceeds slowly but fits each data set to the separate equation. Click **Next** to ensure that the Predicted function results are saved to the worksheet, then **Next** again and make sure no graph is being created. Click **Finish** to end the fit.

To graph the results, you need to create a plot of the predicted results. View the page and select the graph, then click **Add Plot** in the **Graph Additions** group on the **Graph** tab to create a straight line plot of rows 1-12 of column 1 versus rows 1-12 of the predicted results column.

Create two more line plots of rows 13-23 and 24-34 by clicking **Add Plot**. The results plots appear as three separate curves.

Example 7: Advanced Nonlinear Regression

Consider the function:

$$f = 1 - e^{\frac{-dx(b+cx)}{x+a}}$$

When fitted to the data in columns 1 and 2 in the Advanced Techniques worksheet, this equation presents several problems:

- Parameter identifiability.
- Very large x values.
- Very large y error value range.

These problems are outlined and solved below.

If you want to view the regression functions for this equation, open the Advanced Techniques worksheet and graph in the Advanced Techniques section of the Regression Examples (nonlin.jnb) notebook. Double-click the Advanced Techniques Equation to open the Regression Wizard. If you want to run the equation, use the graph of the transformed data.

Overparameterized Equations

The equation has four parameters, *a*, *b*, *c*, and *d*. The numerator in the exponential:

-dx(b+cx)

can have identical values for an infinite number of possible parameter combinations. For example, the parameter values:

b = c = 1 and *d = 2*

and the values:

b = c = 2 and *d = 1*

result in identical numerator terms.

The curve fitter cannot find a unique set of parameters. The parameters are not uniquely *identifiable*, as indicated by the large values for variance inflation factor (VIF), and dependency values near 1.0.

The solution to this problem is to multiply the d parameter with the other terms to create the equation:

$$f = 1 - e^{\frac{-x(db+dcx)}{x+a}}$$

then treat the *db* and *dc* terms as single parameters. This reduces the number of parameters to three.

Scaling Large Variable Values

The data used for the fit have enormous x values, around a value of 1 × 1024 (see column 1 in the worksheet above). These x values appear in the argument of an exponential which is limited to about ±700, which is much smaller than 1024 . However, when the curve fitter tries to find the parameter values which are multiplied with x , it does not try to keep the argument value within ±700. Instead, when the curve fitter varies the parameters, it overflows and underflows the argument range, and does not change the parameter values.

The solution to this problem is to scale the x variable and redefine some of the parameters. Multiply and divide each x value by 1 × 1024 to get:

$$f = 1 - e^{\left[\frac{-\frac{10^{-24}x\left(db+\frac{dc10^{-24}x}{10^{-24}}\right)}{10^{-24}}}{\frac{10^{-24}x}{10^{-24}}+a}\right]}$$

If you let $X = x(10^{-24})$, then the equation becomes:

$$f = 1 - e^{\left[\frac{-X(d\partial + dc10^{24}x)}{X + 10^{24}a}\right]}$$

If you let $CD = 10^{24}dc$ and $A = 10^{-24}a$, the resulting scaled equation is simplified to:

$$f = 1 - e^{\left[\frac{-X(d\partial + CDX)}{X + A}\right]}$$

The exponent argument now does not cause underflows and overflows.

The graph of the transformed x data is displayed below the original data.

Small Independent Variable Values: Weighting for Non-Uniform Errors

The *y* values for the data range from very small values to very large values. However, for this problem, we know that the y values do not have the same errors—smaller y values have smaller errors.

The curve fitter fits the data by minimizing the sum of the squares of the residuals. Because the squares of the residuals extend over an even larger range than the data, small residual squared numbers are essentially ignored.

The solution to this non-uniform error problem is to use weighting, so that all residual squared terms are approximately the same size.

Fitting with a weighting variable of $1/y^2$ (the inverse of y squared), which is proportional to the inverse of the variance of the y data, produces a better fit for low y value data.

To see the results of the regression without weighting, open the Options dialog box and change the weighting to (none) before finishing.

Chapter 9

Transform Function Reference

Topics:

SigmaPlot provides many predefined functions, including arithmetic, statistical, trigonometric, and number-generating functions. In addition, you can define functions of your own.

Function Arguments

Function arguments are placed in parentheses following the function name, separated by commas. Arguments must be typed in the sequence shown for each function.

You must provide the required arguments for each function first, followed by any optional arguments desired. Any omitted optional arguments are set to the default value. Optional arguments are always omitted from right to left. If only one argument is omitted, it will be the last argument. If two are omitted, the last two arguments are set to the default value.

You can use a missing value (i.e., 0/0) as a placeholder to omit an argument.

Example

The col function has three arguments: *column*, *top*, and *bottom*. Therefore, the syntax for the col function is: `col(column,top,bottom)`

The column number argument is required, but the first (top) and last (bottom) rows are optional, defaulting to row 1 as the first row and the last row with data for the last row.

`col(2)` returns the entirety of column 2.

`col(2,5)` returns column 2 from row 5 to the end of the column.

`col(2,5,100)` returns column 2 from row 5 to row 100.

`col(2,0/0,50)` returns column 2 from row 1 to the 50th row in the column.

User-Defined Functions

Important: For a complete list of all User-Defined Transforms, see Transform Function Descriptions on page 161.

You can create any user-defined function, consisting of any expression in the transform language, and then refer to it by name.

For example, the following transform defines the function `dist2pts`, which returns the distance between two points `dist2pts(x1,y1,x2,y2) = sqrt((x2-x1)^2+(y2-y1)^2)`.

You can then use this custom-defined function, instead of the expression to the right of the equal sign, in subsequent equations.

For example, to plot the distances between two sets of XY coordinates, with the first points stored in columns 1 and 2, and the second in columns 3 and 4, enter:

```
col(5) = dist2pts(col(1),col(2),col(3),col(4))
```

The resulting distances are placed in column 5.

Saving User-Defined Functions

Frequently used variable values and custom transforms can be saved to a transform file, then copied and pasted into the desired transform.

To save user-defined functions to a file, then apply them to a transform:

1. Define the variables and functions in the **Transform** window, then click **Save**.
2. When the **Save** dialog box appears, name the file something like `"User-Defined Functions"`.
3. Select the function you want to use in the transform, then press **Ctrl+C**.

4. Open the transform file you want to copy the function to, click the point in the text where you want to enter the function, then press **Ctrl+V**. For more information, see Transform Function Descriptions on page 161.

Transform Function Descriptions

You can modify and manipulate worksheet data by entering SigmaPlot's extensive mathematical transformation language into the **User-Defined Transform** dialog box. Type transform instructions into the **Edit Transform** field. You can enter up to 32,000 characters.

The following list groups transforms by function type. It is followed by an alphabetical reference containing complete descriptions of all transform functions and their syntax, with examples.

Worksheet Functions

These worksheet functions are used to specify cells and columns from the worksheet, either to read data from the worksheet for transformation, or to specify a destination for transform results.

Function	Description
block	The block function returns a specified block of cells from the worksheet.
blockheight, blockwidth	The blockheight and blockwidth functions return a specified block of cells or block dimension from the worksheet.
cell	The cell function returns a specific cell from the worksheet.
col	The col function returns a worksheet column or portion of a column.
put into	The put into function places variable or equation results in a worksheet column.
subblock	The subblock function returns a specified block of cells from within another block.

Data Manipulation Functions

The data manipulation functions are used to generate non-random data, and to sample, select, and sort data.

Function	Description
data	The data function generates serial data.
if	The if function conditionally selects between two data sets.
nth	The nth function returns an incremental sampling of data.
sort	The sort function rearranges data in ascending order.

Matrix Functions

These functions compute common operations for matrices, as well as providing solutions to systems of linear equations, multiple linear regression problems, and the eigenvalue problem for symmetric matrices.

Function	Description
inv	The inv function generates the inverse matrix of an invertible square matrix provided as a block.
trp	The trp function generates the transpose of any matrix provided as a block. The entries of the matrix can be non-numeric.

prod	The prod function computes the product of two matrices where each matrix is provided as a block or a range. The product is defined as usual matrix multiplication where entries in the result are computed as the inner products of rows of the first matrix and columns of the second matrix.
eigen	The eigen function computes the eigenvalues and corresponding eigenvectors for a real symmetric matrix that is provided as a block.
linsys	The linsys function solves systems of linear equations and multiple linear regression problems.

Trigonometric Functions

SigmaPlot provides a complete set of trigonometric functions.

Function	Description
arccos	This functions returns the arccosine, of the specified argument.
arcsin	This functions returns the arcsine of the specified argument.
arctan	This functions returns the arctangent of the specified argument.
cos	This function returns the cosine of the specified argument.
sin	This function returns the sine of the specified argument.
tan	This function returns the tangent of the specified argument.
cosh	This function returns the hyperbolic cosine of the specified argument.
sinh	This function returns the hyperbolic sine of the specified argument.
tanh	This function returns the hyperbolic tangent of the specified argument.

Numeric Functions

The numeric functions perform a specific type of calculation on a number or range of numbers and returns the appropriate results.

Function	Description
abs	The abs function returns the absolute value.
exp	The exp function returns the values for e raised to the specified numbers.
factorial	The factorial function returns the factorial for each specified number.
mod	The mod function returns the modulus, or remainder of division, for specified numerators and divisors.
ln	The ln function returns the natural logarithm for the specified numbers.
log	The log function returns the base 10 logarithm for the specified numbers.
sqrt	The sqrt function returns the square root for the specified numbers.

Range Functions

The following functions give information on ranges.

Function	Description
count	The count function returns the number of numeric values in a range.

makeblock	The makeblock function returns a range or block of data having a specified size using a range of data that you supply.
missing	The missing function returns the number of missing values and text strings in a range.
size	The size function returns the number of data points in a range, including all numbers, missing values, and text strings.

Accumulation Functions

The accumulation functions return values equal to the accumulated operation of the function.

Function	Description
diff	The diff function returns the differences of the numbers in a range.
sum	The sum function returns the cumulative sum of a range of numbers.
total	The total function returns the value of the total sum of a range.

Random Generation Functions

You can use the two “random” number generating functions to create a series of normally or uniformly distributed numbers.

Function	Description
gaussian	The Gaussian function is used to generate a series of normally (Gaussian or “bell” shaped) distributed numbers with a specified mean and standard deviation.
random	The random function is used to generate a series of uniformly distributed numbers within a specified range.

Precision Functions

Use the precision functions to convert numbers to whole numbers or to round off numbers.

Function	Description
int	The int function converts numbers to integers.
prec	The prec function rounds numbers off to a specified number of significant digits.
round	The round function rounds numbers off to a specified number of decimal places.

Statistical Functions

The statistical functions perform statistical calculations on a range or ranges of numbers.

Function	Description
avg	The avg function calculates the averages of corresponding numbers across ranges. It can be used to calculate the average across rows for worksheet columns.
max	The max function returns the largest value in a range.
min	The min function returns the smallest value in a range.
mean	The mean function calculates the mean of a range.
median	The median function calculates the median of a range.
runavg	The runavg function produces a range of running averages.
stddev	The stddev function returns the standard deviation of a range.

stderr	The stderr function calculates the standard error of a range.

Area and Distance Functions

Use these functions to calculate the areas and distances specified by X,Y coordinates. Units are based on the units used for X and Y.

Function	Description
area	The area function finds the area of a polygon described in X,Y coordinates.
dist	The distance function calculates the distance of a line whose segments are described in X,Y coordinates.
partdist	The partdist function calculates the distances from an initial X,Y coordinate to successive X,Y coordinates in a cumulative fashion.

Curve Fitting Functions

These functions are designed to be used in conjunction with SigmaPlot's nonlinear curve fitter, to allow automatic determination of initial equation parameter estimates from the source data. You can use these functions to develop your own parameter determination function by using the functions provided with the Standard Regression Equations library provided with SigmaPlot.

Function	Description
ape	This function is used for the polynomials, rational polynomials and other functions which can be expressed as linear functions of the parameters. A linear least squares estimation procedure is used to obtain the parameter estimates.
dsinp	This function returns an estimate of the phase in radians of damped sine functions.
fwhm	This function returns the x width of a peak at half the peak's maximum value for peak shaped functions.
lowess	The lowess algorithm is used to smooth noisy data. "Lowess" means *locally weighted regression*. Each point along the smooth curve is obtained from a regression of data points close to the curve point with the closest points more heavily weighted.
lowpass	The lowpass function returns smoothed y values from ranges of x and y variables, using an optional user-defined smoothing factor that uses FFT and IFFT.
sinp	This function returns an estimate of the phase in radians of sinusoidal functions.
x25	This function returns the x value for the y value 25% of the distance from the minimum to the maximum of smoothed data for sigmoidal shaped functions.
x50	This function returns the x value for the y value 50% of the distance from the minimum to the maximum of smoothed data for sigmoidal shaped functions.
x75	This function returns the x value for the y value 75% of the distance from the minimum to the maximum of smoothed data for sigmoidal shaped functions.
xatymax	This function returns the x value for the maximum y in the range of y coordinates for peak shaped functions.
xwtr	This function returns x75-x25 for sigmoidal shaped functions.

Fast Fourier Transform Functions

Use these functions to remove noise from and smooth data using frequency-based filtering.

Function	Description

fft	The fft function finds the frequency domain representation of your data.
invfft	The invfft function takes the inverse fft of the data produced by the fft to restore the data to its new filtered form.
real	The real function strips the real numbers out of a range of complex numbers.
img	The img function strips the imaginary numbers out of a range of complex numbers.
complex	The complex function converts a block of real and/or imaginary numbers into a range of complex numbers.
mulcpx	The mulcpx function multiplies two ranges of complex numbers together.
invcpx	The invcpx takes the reciprocal of a range of complex numbers.

Probability Functions

Use these functions to compute and verify statistical measures such as significant probabilities, critical values of statistics, confidence intervals and histogram comparisons.

Function	Description
normdist	This function is the cumulative normal (or Gaussian) distribution function. It returns the probability that a normal random variable is less than a specified independent variable value.
norminv	This function is the inverse cumulative normal (or Gaussian) distribution function. The probability that a normally distributed random variable is less than the return value is equal to the argument you specify.
normden	This function is the normal (or Gaussian) probability density function. The graph of this function is the familiar “bell curve”. It returns the value of the slope of the cumulative distribution function at the specified argument value.
chisquaredist	This function is the cumulative chi-square distribution function. It returns the probability that a chi-square distributed random variable is less than a specified independent variable value.
chisquareinv	This function is the inverse cumulative chi-square distribution function. The probability that a chi-square distributed random variable is less than the return value is equal to the argument you specify.
chisquareden	This function is the chi-square probability density function. It returns the value of the slope of the cumulative distribution function at the specified argument value.
tdist	This function is Student’s T-distribution function. It returns the probability that a T-distributed random variable is less than a specified independent variable value.
tinv	This function is the inverse of Student’s T-distribution function. The probability that a T-distributed random variable is less than the return value is equal to the argument you specify.
tden	This function is the T-distribution’s probability density function. It returns the value of the slope of the cumulative distribution function at the specified argument value.

fdist	This function is the F-distribution function. It returns the probability that an F distributed random variable is less than a specified independent variable value.
finv	This function is the inverse F-distribution function. The probability that an F-distributed random variable is less than the return value is equal to the argument you specify.
fden	This function is the F-distribution's probability density function. It returns the value of the slope of the cumulative distribution function at the specified argument value.
gammadist	This function is the cumulative gamma distribution function. It returns the probability that a gamma distributed random variable is less than a specified independent variable value.
gammainv	This function is the inverse cumulative gamma distribution function. The probability that a gamma distributed random variable is less than the return value is equal to the argument you specify.
gammaden	This function is the gamma distribution's probability density function. It returns the value of the slope of the cumulative distribution function at the specified argument value.
weibulldist	This function is the cumulative Weibull distribution function. It returns the probability that a Weibull distributed random variable is less than a specified independent variable value.
weibullinv	This function is the inverse cumulative Weibull distribution function. The probability that a Weibull distributed random variable is less than the return value is equal to the argument you specify.
weibullden	This function is the Weibull distribution's probability density function. It returns the value of the slope of the cumulative distribution function at the specified argument value.
cauchydist	This function is the cumulative Cauchy distribution function. It returns the probability that a Cauchy distributed random variable is less than a specified independent variable value.
cauchyinv	This function is the inverse cumulative Cauchy distribution function. The probability that a Cauchy distributed random variable is less than the return value is equal to the argument you specify.
cauchyden	This function is the Cauchy distribution's probability density function. It returns the value of the slope of the cumulative distribution function at the specified argument value.
erf	This function is the error function. It is related to the cumulative normal distribution function by scaling and translation.
erfc	This function is the complementary error function. It returns one minus the return value of the error function.

lognormdist	This function is the cumulative log-normal distribution function. It returns the probability that a log-normal random variable is less than a specified independent variable value.
lognorminv	This function is the inverse cumulative log-normal distribution function. The probability that a log-normal random variable is less than the return value is equal to the argument you specify.
lognormden	This function is the log-normal distribution's probability density function. It returns the value of the slope of the cumulative distribution function at the specified argument value.
expdist	This function is the cumulative exponential distribution function. It returns the probability that an exponential random variable is less than a specified independent variable value.
expinv	This function is the inverse cumulative exponential distribution function. The probability that an exponential random variable is less than the return value is equal to the argument you specify.
expden	This function is the exponential distribution's probability density function. It returns the value of the slope of the cumulative distribution function at the specified argument value.
logisdist	This function is the cumulative logistic distribution function. It returns the probability that a logistic random variable is less than a specified independent variable value.
logisinv	This function is the inverse cumulative logistic distribution function. The probability that a logistic random variable is less than the return value is equal to the argument you specify.
logisden	This function is the logistic distribution's probability density function. It returns the value of the slope of the cumulative distribution function at the specified argument value.
loglogisdist	This function is the cumulative log-logistic distribution function. It returns the probability that a log-logistic random variable is less than a specified independent variable value.
loglogisinv	This function is the inverse cumulative log-logistic distribution function. The probability that a log-logistic random variable is less than the return value is equal to the argument you specify.
loglogisden	This function is the log-logistic distribution's probability density function. It returns the value of the slope of the cumulative distribution function at the specified argument value.

Special Constructs

Transform constructs are special structures that allow more complex procedures than functions.

Function	Description
for	The for statement is a looping construct used for iterative processing.

if...then...else	The if...then...else construct proceeds along one of two possible series of procedures based on the results of a specified condition.

Miscellaneous Functions

These functions are specialized functions which perform a variety of operations.

Function	Description
choose	The choose function is the mathematical “n choose r” function.
histogram	The histogram function generates a histogram from a range or column of data.
implicit	The implicit function finds the values of a function defined implicitly by an equation that includes one or more independent variables, each have a specified range of values.
interpolate	The interpolate function performs linear interpolation between X,Y coordinates.
lookup	The lookup function compares values with a specified table of boundaries and returns either a corresponding index from a one-dimensional table, or a corresponding value from a two-dimensional table.
polynomial	The polynomial function returns results for specified independent variables for a specified polynomial equation.
rgbcolor	The rgbcolor(r,g,b) color function takes arguments r,g, and b between 0 and 255 and returns color to cells in the worksheet.
root	The root function finds the roots or singularities of one or more functions of one variable over a finite interval and returns the range of values found.

abs

The *abs* function returns the absolute value for each number in the specified range.

Syntax

```
abs(numbers)
```

The numbers argument can be a scalar or range of numbers. Any missing value or text string contained within a range is ignored and returned as the string or missing value.

Example

The operation `col(2) = abs(col(1))` places the absolute values of the data in column 1 in column 2.

ape

The *ape* function is used for the polynomials, rational polynomials and other functions which can be expressed as linear functions of the parameters. A linear least squares estimation procedure is used to obtain the parameter estimates. The ape function is used to automatically generate the initial parameter estimates for SigmaPlot’s nonlinear curve fitter from the equation provided.

Syntax

```
ape(x range,y range,n,m,s,f)
```

The *x range* and *y range* arguments specify the independent and dependent variables, or functions of them (for example, ln(x)). Any missing value or text string contained within one of the ranges is ignored and will not be treated as a data point. *x range* and *y range* must be the same size.

The *n* argument specifies the order of the numerator of the equation. The *m* argument specifies the order of the denominator of the equation. *n* and *m* must be greater than or equal to 0 (*n*, *m*, ≥ 0). If *m* is greater than 0 then n must be less than or equal to *m* (if $m > 0$, $n \leq m$).

The *s* argument specifies whether or not a constant is used. *s*=0 specifies no constant term y0 in the numerator, *s*=1 specifies a constant term y_0 in the numerator. *s* must be either 0 or 1. *If n* = 0, *s* cannot be 0 (there must be a constant).

The number of valid data points must be greater than or equal to $n = m = s$.

The optional *f* argument defines the amount of Lowess smoothing, and correspond ins

Example

The operation `col(2) = asin(col(1))` places the arcsine of all column 1 data points in column 2.

arctan

This function returns the inverse of the corresponding trigonometric function.

Syntax

```
arctan(numbers)
```

The *numbers* argument can be a scalar or range. You can also use the abbreviated function name atan.

The numbers argument can be any value. Results are returned in degrees, radians, or grads, depending on the **Trigonometric Units** selected in the **User-Defined Transform** dialog box.

The function range (in radians) is:

$$\arctan \quad -\frac{\pi}{2} to \frac{\pi}{2}$$

Example

The operation `col(2) = atan(col(1))` places the arctangent of all column 1 data points in column 2.

Tip: A convenient way of obtaining the value of π is π = *4 + atan(1*).

area

The area function returns the area of a simple polygon. The outline of the polygon is formed by the xy pairs specified in an x range and a y range. The list of points does not need to be closed. If the last xy pair does not equal the first xy pair, the polygon is closed from the last xy pair to the first. The area function only works with simple non-overlapping polygons. If line segments in the polygon cross, the overlapping portion is considered a negative area, and results are unpredictable.

Syntax

```
area(x range,y range)
```

The *x range* argument contains the *x* coordinates, and the *y range* argument contains the *y* coordinates. Corresponding values in these ranges form xy pairs.

If the ranges are uneven in size, excess *x* or *y* points are ignored.

Example

For the ranges *x* = {0,1,1,0} and *y* = {0,0,1,1}, the operation area (*x,y*) returns a value of 1. The *x* and *y* coordinates provided describe a square of 1 unit.

avg

The avg function averages the numbers across corresponding ranges, instead of within ranges. The resulting range is the row-wise average of the range arguments. Unlike the mean function, avg returns a range, not a scalar.

The avg function calculates the arithmetic mean, defined as:

$$x = \frac{1}{n}\sum_{i=1}^{n} x_i$$

The avg function can be used to calculate averages of worksheet data across rows rather than within columns.

Syntax

```
avg({x1,x2...},{y1,y2...},{z1,z2...})
```

The x_1, y_1, and z_1 are corresponding numbers within ranges. Any missing value or text string contained within a range returns the string or missing value as the result.

Example

The operation `avg({1,2,3},{3,4,5})` returns {2,3,4}. 1 from the first range is averaged with 3 from the second range, 2 is averaged with 4, and 3 is averaged with 5. The result is returned as a range.

block

The block function returns a block of cells from the worksheet, using a range specified by the upper left and lower right cell row and column coordinates.

Syntax

```
block(column 1,row 1,column 2,row 2)
```

The *column 1* and *row 1* arguments are the coordinates for the upper left cell of the block; the *column 2* and *row 2* arguments are the coordinates for the lower right cell of the block. All values within this range are returned. Operations performed on a block always return a block.

If *column 2* and *row 2* are omitted, then the last row and/or column is assumed to be the last row and column of the data in the worksheet. If you are equating a block to another block, then the last row and/or column is assumed to be the last row and column of the equated block (see the following example).

All column and row arguments must be scalar (not ranges). To use a column title for the column argument, enclose the column title in quotes; block uses the column in the worksheet whose title matches the string.

Example

The command `block(5,1) = -block(1,1,3,24)` reverses the sign for the values in the range from cell (1,1) to cell (3,24) and places them in a block beginning in cell (5,1).

block

The block function returns a block of cells from the worksheet, using a range specified by the upper left and lower right cell row and column coordinates.

Syntax

```
block(column 1,row 1,column 2,row 2)
```

The *column 1* and *row 1* arguments are the coordinates for the upper left cell of the block; the *column 2* and *row 2* arguments are the coordinates for the lower right cell of the block. All values within this range are returned. Operations performed on a block always return a block.

If *column 2* and *row 2* are omitted, then the last row and/or column is assumed to be the last row and column of the data in the worksheet. If you are equating a block to another block, then the last row and/or column is assumed to be the last row and column of the equated block (see the following example).

All column and row arguments must be scalar (not ranges). To use a column title for the column argument, enclose the column title in quotes; block uses the column in the worksheet whose title matches the string.

Example

The command `block(5,1) = -block(1,1,3,24)` reverses the sign for the values in the range from cell (1,1) to cell (3,24) and places them in a block beginning in cell (5,1).

blockheight, blockwidth

The blockheight and blockwidth functions return the number of rows or columns, respectively, of a defined block of cells from the worksheet.

Syntax

```
blockheight(block) blockwidth(block)
```

The block argument can be a variable defined as a block, or a block function statement.

Example

For the statement `x = block(2,1,12,10) :`

The operation `cell(1,1) = blockheight(x)` places the number 10 in column 1, row 1 of the worksheet.

The operation `cell(1,2) = blockwidth(x)` places the number 11 in column 1, row 2 of the worksheet.

cauchyden

This function is the Cauchy distribution's probability density function. It returns the value of the slope of the cumulative distribution function at the specified argument value.

Syntax

```
cauchyden(x,a,b)
```

The x argument represents the independent variable and can either be a scalar or a range of numbers. If x is a range, then it must be defined by either using braces { } or by specifying a worksheet column. Any value for x must be real. The a argument is any real number and is the location parameter, equal to the location of the density function's peak. The b argument is any positive number and is the scale parameter, equal to the half-width at half-maximum of the density function's peak.

Example

The density function can be used to estimate the probability that the values of a Cauchy distributed random variable C lie in a small interval. If C has a peak location equal to 3 and a half-width at half-maximum equal to 2, then to estimate the probability that the values of C lie between 2 and 2.1, multiply the density of C at 2 by the length of the interval .1:

```
cauchyden(2,3,2) * .1 = .012732
```

cauchydist

This function is the cumulative Cauchy distribution function. It returns the probability that a Cauchy distributed random variable is less than a specified independent variable value.

A Cauchy distributed random variable is the distribution of the ratio of a two normal random variables. It is also the distribution of the random variable Y = tan(X), where X is uniformly distributed.

This distribution is used to describe forced resonance behavior and the shape of spectral lines subject to homogenous broadening.

Syntax

```
cauchydist(x,a,b)
```

The x argument represents the independent variable and can either be a scalar or a range of numbers. If x is a range, then it must be defined by either using braces { } or by specifying a worksheet column. Any value for x must be real.

The a argument is any real number and is the location parameter. The b argument is any positive number and is the shape parameter.

Example

Suppose a Cauchy distributed random variable C has a location parameter equal to 3 and a shape parameter equal to 3. To compute the probability that the values of C exceed 2, we calculate:

```
P( C > 2 ) = 1 – P( C < 2 ) = 1 – cauchydist(2,3,2) = .64758
```

cauchyinv

This function is the inverse cumulative Cauchy distribution function. The probability that a Cauchy distributed random variable is less than the return value is equal to the argument you specify.

Syntax

```
cauchyinv(x,a,b)
```

The *x* argument can either be a scalar or a range of numbers. If *x* is a range, then it must be defined by either using braces { } or by specifying a worksheet column. Any scalar value for x represents a probability and so must be between 0 and 1. The a argument is any real number and is the location parameter. The *b* argument is any positive number and is the shape parameter.

Example

Suppose a Cauchy distributed random variable C has a location parameter equal to 3 and a shape parameter equal to 2. To calculate the tail of this distribution whose probability is .05, we need to find a number c such that P(C > c) = .05. This is the same as finding c such that P(C < c) = .95. Therefore, we calculate:

```
cauchyinv(.95, 3, 2) = 15.62750
```

cell

The cell function returns the contents of a cell in the worksheet, and can specify a cell destination for transform results.

Syntax

```
cell (column,row)
```

Both column and row arguments must be scalar (not ranges). To use a column title for the column argument, enclose the column title in quotes; cell uses the column in the worksheet whose title matches the string.

Data placed in a cell inserts or overwrites according to the current insert mode.

Example

This example shows how you can use the cell function to label column and row titles with the results of a transform.

For example, take `cell (2,3)= "dog"`. In this case, the function places the text string "dog" (without quotes) into the second column, third row of the worksheet.

But if you were to replace the 2 with a 0, as in `cell (0,3)= "cat"`, this changes the title of the third row of the worksheet to "cat".

Likewise, if you were to change the 3 to 0, as in `cell (2,0)= "parrot"`, this changes the title of the second column to "parrot."

chisquareden

This function is the chi-square probability density function. It returns the value of the slope of the cumulative distribution function at the specified argument value.

Syntax

```
chisquareden(x,n)
```

The *x* argument represents the independent variable and can either be a scalar or a range of numbers. If *x* is a range, then it must be defined by either using braces { } or by specifying a worksheet column. Any value for *x* must be non-negative. The *n* argument can be any positive integer and equals the degrees of freedom.

Example

The density function can be used to estimate the probability that the values of a chi-square distributed random variable X lie in a small interval. If X has 8 degrees of freedom, then to estimate the probability that the values of X lie between 5 and 5.1, multiply the density of X at 5 by the length of the interval .1:

```
chisquareden(5,8) * .1 = .10688
```

chisquaredist

This function is the cumulative chi-square distribution function. It returns the probability that a chi-square distributed random variable is less than a specified independent variable value.

A chi-square random variable is defined as a sum of squares of independent standard normal distribution variables. The number of normal variables in the sum is called the degrees of freedom.

This distribution is used in goodness-of-fit measures. It describes the distribution of sample variance for a set of normally distributed observations and describes the distribution of the residual sum of squares in regression.

Syntax

```
chisquaredist(x,n)
```

Example

Suppose a random variable X is chi-square distributed with 11 degrees of freedom. To compute the probability that the values of this variable are less than 5, we calculate:

```
chisquaredist(5,11) = .06883
```

chisquareinv

This function is the inverse cumulative chi-square distribution function. The probability that a chi-square distributed random variable is less than the return value is equal to the argument you specify.

Syntax

```
chisquareinv(x,n)
```

The *x* argument can either be a scalar or a range of numbers. If *x* is a range, then it must be defined by either using braces { } or by specifying a worksheet column. Any scalar value for *x* represents a probability and so must be between 0 and 1. The *n* argument can be any positive integer and equals the degrees of freedom.

Example

Suppose a chi-square distributed random variable X has 19 degrees of freedom. To compute the median of X, we calculate:

```
chisquareinv(.5,19) = 18.33765
```

choose

The choose function determines the number of ways of choosing r objects from n distinct objects without regard to order.

Syntax

```
choose(n,r)
```

For the arguments n and r, r < n and “n choose r” is defined as:

$$\binom{n}{r} = \frac{n!}{r!(n-r)!}$$

Example

To create a function for the binomial distribution, enter the equation:

```
binomial(p,n,r) = choose(n,r) * (p^r) * (1-p) ^ (n-r)
```

col

The col function returns all or a portion of a worksheet column, and can specify a column destination for transform results.

Syntax

```
col (column,top,bottom)
```

The *column* argument is the column number or title. To use a column title for the column argument, enclose the title in quotation marks. The *top* and *bottom* arguments specify the first and last row numbers, and can be omitted. The default row numbers are 1 and the end of the column, respectively; if both are omitted, the entire column is used. All parameters must be scalar. Data placed in a column inserts or overwrites according to the current insert mode.

complex

Converts a block of real and imaginary numbers into a range of complex numbers.

Syntax

```
complex (range,range)
```

The first range contains the real values, the second range contains the imaginary values and is optional. If you do not specify the second range, the complex transform returns zeros for the imaginary numbers. If you do specify an imaginary range, it must contain the same number of values as the real value range.

Example

If `x = {1,2,3,4,5,6,7,8,9,10}`, the operation `complex(x)` returns `{{1,2,3,4,....,9,10}, {0,0,0,0,....,0,0}}`.

If `x = {1.0,-0.75,3.1}` and `y = {1.2,2.1,-1.1}`, the operation `complex(x,y)` returns `{{1.0,-0.75,3.1}, {1.2,2.1,-1.1}}`.

cos

This function returns ranges consisting of the cosine of each value in the argument given. This and other trigonometric functions can take values in radians, degrees, or grads. This is determined by the Trigonometric Units selected in the User-Defined Transform dialog box.

Syntax

```
cos(numbers)
```

The *numbers* argument can be a scalar or range.

If you regularly use values outside of the usual -2π to 2π (or equivalent) range, use the mod function to prevent loss of precision. Any missing value or text string contained within a range is ignored and returned as the string or missing value.

Example

If you choose **Degrees** as your Trigonometric Units in the User-Defined Transform dialog box, the operation `cos({0,60,90,120,180})` returns values of {1,0.5,0,-0.5,-1}.

cosh

This function returns the hyperbolic cosine of the specified argument.

Syntax

```
cosh(numbers)
```

The numbers argument can be a scalar or range.

Any missing value or text string contained within a range is ignored and returned as the string or missing value.

Example

The operation `x = cosh(col(2))` sets the variable x to be the hyperbolic cosine of all data in column 2.

count

The count function returns the value or range of values equal to the number of non-missing numeric values in a range. Missing values and text strings are not counted.

Syntax

```
count(range)
```

The *range* argument must be a single range (indicated with the {} brackets) or a worksheet column.

data

The data function generates a range of numbers from a starting number to an end number, in specified increments.

Syntax

```
data(start,stop,step)
```

All arguments must be scalar. The start argument specifies the beginning number and the end argument sets the last number. If the step parameter is omitted, it defaults to 1. The start parameter can be more than or less than the stop parameter. In either case, data steps in the correct direction. Remainders are ignored.

Example

The operation `data(1,5)` returns the range of values {1,2,3,4,5}.

The operation `data(10,1,2)` returns the values {10,8,6,4,2}.

Note: If start and stop are equal, this function produces a number of copies of start equal to step. For example, the operation `data(1,1,4)` returns {1,1,1,1}.

diff

The diff function returns a range or ranges of numbers which are the differences between a given number in a range and the preceding number. The value of the preceding number is subtracted from the value of the following number.

Because there is no preceding number for the first number in a range, the value of the first number in the result is always the same as the first number in the argument range.

Syntax

```
diff(range)
```

The *range* argument must be a single range (indicated with the {} brackets) or a worksheet column. Any missing value or text string contained within the range is returned as the string or missing value.

Example

For x = {9,16,7}, the operation `diff(x)` returns a value of {9,7,-9}.

For y = {4,-6,12}, the operation `diff(y)` returns a value of {4,-10,18}.

dist

The dist function returns a scalar representing the distance along a line. The line is described in segments defined by the X,Y pairs specified in an x range and a y range.

Syntax

```
dist(x range,y range)
```

The x range argument contains the X coordinates, and the y range argument contains the Y coordinates. Corresponding values in these ranges form X,Y pairs. If the ranges are uneven in size, excess X or Y points are ignored.

Example

For the ranges x ={0,1,1,0,0} and y = {0,0,1,1,0}, the operation `dist(x,y)` returns 4.0. The X and Y coordinates provided describe a square of 1 unit x by 1 unit y.

dsinp

The dsinp function automatically generates the initial parameter estimates for a damped sinusoidal functions using the FFT method. The four parameter estimates are returned as a vector.

Syntax

```
dsinp(x range, y range)
```

The *x range* argument specifies the x variable, and the y range argument specifies the y variable. Any missing value or text string contained within one of the ranges is ignored and will not be treated as a data point. *x range* and *y range* must be the same size, and the number of valid data points must be greater than or equal to 3.

Note: *dsinp* is especially used to estimate parameters on waveform functions. This is only useful when this function is used in conjunction with nonlinear regression.

eigen

The eigen function computes the eigenvalues and corresponding eigenvectors for a real symmetric matrix that is provided as a block.

Syntax

```
eigen (block, add eigenvectors)
```

An eigenvalue for any square matrix A is a number that if subtracted from the main diagonal entries of A will result in a singular matrix. If λ is an eigenvalue of A, then a corresponding eigenvector x is any non-zero column vector such that Ax = λx.

The *block* argument represents a symmetric matrix. Since a symmetric matrix is defined as a matrix which equals its transpose, a symmetric matrix is square with an equal number of rows and columns. If the matrix is not square or not symmetric, then an error message is displayed and transform execution stops.

The add *eigenvectors* argument is a number and is optional. If this argument is not provided or has a zero value, then the function will return a range of the eigenvalues of the matrix in decreasing order. If this argument equals any non-zero value, then both eigenvalues and corresponding eigenvectors are returned as a block. This block has *n* rows and *n + 1* columns, where *n* is the number of rows in the symmetric matrix. The first column of the block contains the eigenvalues. The remaining columns contain *n* mutually orthogonal unit eigenvectors of the matrix, where the k^{th} eigenvector corresponds to the k^{th} eigenvalue in the first column.

Example 1

For the matrix:

1.00	2.00	0.00
2.00	1.00	2.00
0.00	2.00	300.

in `block (1,1,3,3)`, the operation

```
block (4,1) = eigen (block (1,1,3,3),1)
```

returns the eigenvalues and eigenvectors of the matrix in `block (4,1,7,3)`:

4.6039	-0.3280	-0.7370	-0.5910
1.8901	-0.5910	-0.3280	-0.7370
-1.4940	-0.7370	-0.5910	-0.3280

The first column lists the eigenvalues in decreasing order and the remaining columns are the corresponding eigenvectors.

You can verify these results using the defining equation $Ax = \lambda x$. For example, using the first eigenvector (the second column of the above matrix) as *x*, the left side of the equation *Ax* is obtained with the prod function:

```
col (8) = prod (block (1,1,3,3), block (5,1,5,3))
```

The right side of the equation λx is obtained by simply multiplying the first eigenvalue by the first eigenvector:

```
col (9) = cell (4,1) * block (5,1,5,3)
```

Both col (8) and col (9) should be equal to high precision.

Example 2

Consider the function *f(x,y)*=-1+4(*e*x-*x*)-5*x*sin *y* +6y^2. This function has a stationary point (a point where to two first partial derivatives are zero) at $x = 0$, $y = 0$. We want to determine whether this point corresponds to a local minimum, local maximum, or a saddle point. To decide this, you first compute the matrix of second partial derivatives (the Hessian) and evaluate this matrix at the point $x = 0$, $y = 0$:

$$\begin{pmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{pmatrix}_{\substack{x=0 \\ y=0}} = \begin{pmatrix} 4e^x & -5\cos y \\ -5\cos y & 5x\sin y + 12 \end{pmatrix}_{\substack{x=0 \\ y=0}} = \begin{pmatrix} 4 & -5 \\ -5 & 12 \end{pmatrix}$$

The type of point at $x = 0$, $y = 0$ can be determined by looking at the two eigenvalues of this last matrix. If both are positive, then we have a local minimum. If both are negative, then we have a local maximum. If one is positive and the other negative, we have a saddle point. Otherwise, results from the eigenvalues are inconclusive.

Enter this matrix:

4	-5
-5	12

in block (1,1,2,2). The operation

```
block (3,1) = eigen (block (1,1,2,2))
```

returns the eigenvalues in block (3,1,3,2):

Table 1:

14.4031
1.5969

These values show that the function has a local minimum at the point $x = 0$, $y = 0$.

erf

This function is the Gauss error function, defined as:

$$erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt.$$

Syntax

```
erf(x)
```

The x argument represents the independent variable and can either be a scalar or a range of numbers. If x is a range, then it must be defined by either using braces { } or by specifying a worksheet column. Any value for x must be real.

Example

If a series of measurements is described by a normal distribution with standard deviation and expected value 0, then the probability that the error of a single measurement is between -.2 and +.2 is given by:

```
erf(.2/(2*sqrt(2)) = .079656
```

erfc

This function is the complementary error function, equal to 1 minus the Gauss error function:

$$erf(x) = 1 - erf(x)$$
$$= \frac{2}{\sqrt{\pi}} \int_{x}^{\infty} e^{-t^2} dt.$$

Syntax

```
erfc(x)
```

The x argument represents the independent variable and can either be a scalar or a range of numbers. If x is a range, then it must be defined by either using braces { } or by specifying a worksheet column. Any value for x must be real.

Example

If a series of measurements is described by a normal distribution with standard deviation # and expected value 0, then the probability that the error of a single measurement is less than -.2 or greater than +.2 is given by:

```
erfc(.2/(2*sqrt(2)) = .920344
```

exp

The exp function returns a range of values consisting of the number e raised to each number in the specified range. This is numerically identical to the expression e^(*numbers*).

Syntax

```
exp(numbers)
```

The *numbers* argument can be a scalar or range of numbers. Any missing value or text string contained within a range is ignored and returned as the string or missing value.

Example

The operation `exp(1)` returns a value of 2.718281828459045.

expden

This function is the exponential distribution's probability density function. It returns the value of the slope of the cumulative distribution function at the specified argument value.

Syntax

```
expden(x,a)
```

The x argument represents the independent variable and can either be a scalar or a range of numbers. If x is a range, then it must be defined by either using braces { } or by specifying a worksheet column. Any value for x must be real. The a argument is any positive number and is the scale parameter.

Example

The density function can be used to estimate the probability that the values of an exponentially distributed random variable X lie in a small interval. If X has scale parameter equal to 3, then to estimate the probability that the values of X lie between 1 and 1.1, multiply the density of X at 1 by the length of the interval .1:

```
expden(1,3) * .1 = .014936
```

expdist

This function is the cumulative exponential distribution function. It returns the probability that an exponential random variable is less than a specified independent variable value.

An exponential distribution is the distribution of time until the first occurrence in a Poisson process, in which events occur continuously and independently at a constant average rate. It is a special case of the Gamma distribution.

Syntax

```
expdist(x,a)
```

The x argument represents the independent variable and can either be a scalar or a range of numbers. If x is a range, then it must be defined by either using braces { } or by specifying a worksheet column. Any value for x must be real. The a argument is any positive number and is the scale parameter.

Example

Suppose an exponential random variable X has scale parameter equal to 3. To compute the probability that the values of X exceed 1, we calculate:

```
P( X > 1 ) = 1 – P( X < 1 ) = 1 – expdist(1,3) = .049787
```

expinv

This function is the inverse cumulative exponential distribution function. The probability that an exponential random variable is less than the return value is equal to the argument you specify.

Syntax

```
expinv(x,a)
```

The x argument can either be a scalar or a range of numbers. If x is a range, then it must be defined by either using braces { } or by specifying a worksheet column. Any scalar value for x represents a probability and so must be between 0 and 1. The a argument is any positive number and is the scale parameter.

Example

Suppose an exponential random variable X has scale parameter equal to 2. To calculate the tail of this distribution whose probability is .05, we need to find a number *x* such that P(X > *x*) = .05. This is the same as finding x such that P(X < *x*) = .95. Therefore, we calculate:

```
expinv(.95, 2) = 1.49787
```

factorial

The factorial function returns the factorial of a specified range.

Syntax

```
factorial({range})
```

The range argument must be a single range (indicated with the {} brackets) or a worksheet column. Any missing value or text string contained within a range is ignored and returned as the string or missing value. Non-integers are rounded down to the nearest integer or 1, whichever is larger.

For factorial(*x*):

$x < 0$ returns a missing value,

$0 \le x < 180$ returns $x!$, and

$x \ge 170$ returns $+\infty$

Example 1

The operation factorial({1,2,3,4,5}) returns {1,2,6,24,120}.

Example 2

To create a transform equation function for the Poisson distribution, you can type:

```
Poisson(m,x)=(m^x)*exp(-m)/factorial(x)
```

fden

This function is the F-distribution's probability density function. It returns the value of the slope of the cumulative distribution function at the specified argument value.

Syntax

```
fden(x,m,n)
```

The *x* argument represents the independent variable and can either be a scalar or a range of numbers. If *x* is a range, then it must be defined by either using braces { } or by specifying a worksheet column. Any value for *x* must be non-negative. The *m* argument is any positive integer and equals the numerator degrees of freedom. The *n* argument is any positive integer and equals the denominator degrees of freedom.

Example

The density function can be used to estimate the probability that the values of an F-distributed random variable F lie in a small interval. If F has a numerator degrees of freedom equal to 3 and a denominator degrees of freedom equal to 14, then to estimate the probability that the values of F lie between 2 and 2.1, multiply the density of F at 2 by the length of the interval .1:

```
fden(2,3,14) * .1 = .014882
```

fdist

This function is the F-distribution function. It returns the probability that an F distributed random variable is less than a specified independent variable value.

An F-distributed random variable is defined as a scaled ratio of a two chi-square variables. The *numerator and denominator degrees of freedom* of an F-distributed variable equal the degrees of freedom of the corresponding chi-square variables.

This distribution is used to test goodness-of-fit in regression problems and for testing the homogeneity of populations for many groups of normally distributed observations.

Syntax

```
fdist(x,m,n)
```

The *x* argument represents the independent variable and can either be a scalar or a range of numbers. If *x* is a range, then it must be defined by either using braces { } or by specifying a worksheet column. Any value for *x* must be non-negative. The *m* argument is any positive integer and equals the numerator degrees of freedom. The *n* argument is any positive integer and equals the denominator degrees of freedom.

Example

Suppose an F-distributed random variable F has a numerator degrees of freedom equal to 3 and a denominator degrees of freedom equal to 14. To compute the probability that the values of F exceed 2, we calculate:

```
P( F > 2 ) = 1 – P( F < 2 ) = 1 – fdist(2,3,14) = .16035
```

fft

The fft function finds the frequency domain representation of your data using the Fast Fourier Transform.

Syntax

```
fft(range)
```

The parameter can be a range of real values or a block of complex values. For complex values there are two columns of data. The first column contains the real values and the second column represents the imaginary values. This function works on data sizes of size 2^n numbers. If your data set is not 2^n in length, the fft function pads 0 at the beginning and end of the data range to make the length 2^n.

The fft function returns a range of complex numbers.

Example

For `x = {1,2,3,4,5,6,7,8,9,10}`, the operation `fft(x)` takes the Fourier transform of the ramp function with real data from 1 to 10 with 3 zeros padded on the front and back and returns a 2 by 16 block of complex numbers.

finv

This function is the inverse F-distribution function. The probability that an F-distributed random variable is less than the return value is equal to the argument you specify.

Syntax

```
finv(x,m,n)
```

The *x* argument can either be a scalar or a range of numbers. If *x* is a range, then it must be defined by either using braces { } or by specifying a worksheet column. Any scalar value for *x* represents a probability and so must be between 0 and 1. The *m* argument is any positive integer and equals the numerator degrees of freedom. The *n* argument is any positive integer and equals the denominator degrees of freedom.

Example

Suppose an F-distributed random variable F has a numerator degrees of freedom equal to 3 and a denominator degrees of freedom equal to 14. To calculate the tail of this distribution whose probability is .05, we need to find a number f such that P(F > f) = .05. This is the same as finding f such that P(F < f) = .95. Therefore, we calculate:

```
finv(.95, 3, 14) = 3.34389
```

for

The for statement is a looping construct used for iterative processing.

Syntax

```
for loop variable = initial value to end value step increment do
equation
equation
.
.
.
end for
```

Transform equation statements are evaluated iteratively within the for loop. When a for statement is encountered, all functions within the loop are evaluated separately from the rest of the transform.

The *loop variable* can be any previously undeclared variable name. The *initial value* for the loop is the beginning value to be used in the loop statements. The *end value* for the loop variable specifies the last value to be processed by the for statement. After the end value is processed, the loop is terminated. In addition, you can specify a loop variable step *increment*, which is used to "skip" values when proceeding from the initial value to end value. If no increment is specified, an increment of 1 is assumed.

Important: You must separate *for, to, step, do, end for*, and all condition statement operators, variables and values with spaces. The for loop statement is followed by a series of one or more transform equations which process the loop variable values.

Inside for loops, you can:

- Indent equations.
- Nest for loops.

Note that these conditions are allowed only within for loops. You cannot redefine variable names within for loops.

Example 1

The operation:

```
for i = 1 to size(col(1)) do
cell(2,i) = cell(1,i)*i
end for
```

multiplies all the values in column 1 by their row number and places them in column 2.

Example 2

The operation:

```
for j = cell(1,1) to
cell (1,64) step 2 do col(10) = col(9)^j
end for
```

takes the value from cell (1,1) and increments by 2 until the value in cell (1,64) is reached, raises the data in column 9 to that power, and places the results in column 10.

fwhm

The fwhm function returns value of the x width at half-maxima in the ranges of coordinates provided, with optional Lowess smoothing.

Syntax

```
fwhm(x range, y range,f)
```

The *x range* argument specifies the x variable, and the y range argument specifies the y variable. Any missing value or text string contained within one of the ranges is ignored and will not be treated as a data point. *x range* and *y range* must have the same size, and the number of valid data points must be greater than or equal to 3.

The optional *f* argument defines the amount of Lowess smoothing, and corresponds to the fraction of data points used for each regression. *f* must be greater than or equal to 0 and less than or equal to 1. $0 \le f \le 1$. If *f* is omitted, no smoothing is used.

Example

For $x = \{0,1,2\}$, $y=\{0,1,4\}$, the operation

```
col(1)=fwhm(x,y)
```

places the *x* width at half-maxima 1.00 into column 1.

gammaden

This function is the gamma distribution's probability density function. It returns the value of the slope of the cumulative distribution function at the specified argument value.

The Gamma and Beta functions occur frequently in many applications. You can compute their values by writing simple transforms expressed in terms of the gamma density function.

The Gamma function can be defined by:

```
Gamma(x) = 1/(exp(1.0)*gammaden(1,x,1))
```

and then the Beta can be defined by:

```
Beta(x,y) = Gamma(x)*Gamma(y)/Gamma(x+y)
```

Syntax

```
gammaden(x,a,b)
```

The x argument represents the independent variable and can either be a scalar or a range of numbers. If x is a range, then it must be defined by either using braces { } or by specifying a worksheet column. Any value for x must be real. The a argument is any positive number and is the shape parameter. The b argument is any positive number and is the scale parameter.

Example

The density function can be used to estimate the probability that the values of a gamma distributed random variable G lie in a small interval. If G has shape parameter equal to 3 and scale parameter equal to 1, then to estimate the probability that the values of G lie between 2 and 2.1, multiply the density of G at 2 by the length of the interval .1:

```
gammaden(2,3,1) * .1 = .027067
```

gammadist

This function is the cumulative gamma distribution function. It returns the probability that a gamma distributed random variable is less than a specified independent variable value.

A gamma distribution is the distribution of time until the nth occurrence in a Poisson process, in which events occur continuously and independently at a constant average rate.

Syntax

```
gammadist(x,a,b)
```

The x argument represents the independent variable and can either be a scalar or a range of numbers. If x is a range, then it must be defined by either using braces { } or by specifying a worksheet column. Any value for x must be real. The a argument is any positive number and is the shape parameter. The b argument is any positive number and is the scale parameter.

Example

Suppose a gamma distributed random variable G has shape parameter equal to 3 and scale parameter equal to 1. To compute the probability that the values of G exceed 2, we calculate:

```
P( G > 2 ) = 1 – P( G < 2 ) = 1 – gammadist(2,3,1) = .67668
```

gammainv

This function is the inverse cumulative gamma distribution function. The probability that a gamma distributed random variable is less than the return value is equal to the argument you specify.

Syntax

```
gammainv(x,a,b)
```

The x argument can either be a scalar or a range of numbers. If x is a range, then it must be defined by either using braces { } or by specifying a worksheet column. Any scalar value for x represents a probability and so must be between 0 and 1. The a argument is any positive number and is the shape parameter. The b argument is any positive number and is the scale parameter.

Example

Suppose a gamma distributed random variable G has shape parameter equal to 3 and scale parameter equal to 1. To calculate the tail of this distribution whose probability is .05, we need to find a number g such that P(G > g) = .05. This is the same as finding g such that P(G < g) = .95. Therefore, we calculate:

```
gammainv(.95, 3, 1) = 6.29579
```

gaussian

This function generates a specified number of normally (Gaussian or “bell” shaped) distributed numbers from a seed number, using a supplied mean and standard deviation.

Syntax

```
gaussian(number,seed,mean,stddev)
```

The *number* argument specifies how many random numbers to generate.

The *seed* argument is the random number generation seed to be used by the function. If you want to generate a different random number sequence each time the function is used, enter 0/0 for the seed. Enter the same number to generate an identical random number sequence. If the seed argument is omitted, a randomly selected seed is used.

The *mean* and *stddev* arguments are the mean and standard deviation of the normal distribution curve, respectively. If mean and stddev are omitted, they default to 0 and 1.

Note that function arguments are omitted from right to left. If you want to specify a stddev, you must either specify the mean argument or omit it by using 0/0.

Example

The operation `gaussian(100)` uses a seed of 0 to produce 100 normally distributed random numbers, with a mean of 0.0 and a standard deviation of 1.0.

histogram

The histogram function produces a histogram of the values range in a specified range, using a defined interval set.

Syntax

```
histogram(range, buckets, interval type)
```

The *range* argument must be a single range (indicated with the {} brackets) or a worksheet column. Any missing value or text string contained within a range is ignored.

The *buckets* argument is used to specify either the number of evenly incremented histogram intervals, or both the number and ranges of the intervals. This value can be scalar or a range. In both versions, missing values and strings are ignored.

The *interval type* argument is an optional value that determines whether the interval for a bucket includes its right endpoint or its left endpoint. Exactly one of the endpoints must be included to guarantee that each data value is unambiguously assigned to a bucket. If the argument is missing or has the value 0, then the right endpoint is included. If the argument has the value 1, then the left endpoint is included.

If the buckets parameter is a scalar, it must be a positive integer. A scalar buckets argument generates a number of intervals equal to the buckets value. The histogram intervals are evenly sized; the range is the minimum value to the maximum value of the specified range.

If the buckets argument is specified as a range, each number in the range becomes the upper bound (inclusive) of an interval. Values from -∞ to ≤ the first bucket fall in the first histogram interval, values from > first bucket to ≤ second bucket fall in the second interval, etcetera. The buckets range must be strictly increasing in value. An additional interval is defined to catch any value which does not fall into the defined ranges. The number of values occurring in this extra interval (including 0, or no values outside the range) becomes the last entry of the range produced by histogram function.

1

For col(1) = {1,20,30,35,40,50,60}, the operation col(2) = histogram(col(1),3) places the range {2,3,2} in column 2. The bucket intervals are automatically set to 20, 40, and 60, so that two of the values in column 1 fall under 20, three fall under 40, and two fall under 60.

2

For buckets = {25,50,75}, the operation `col(3) = histogram(col(1),buckets)` places {2,4,1,0} in col(3). Two of the values in column 1 fall under 25, four fall under 50, one under 75, and no values fall outside the range.

3

For buckets = {25,50,75} and using the column 1 data in Example 1, the operation col(3) = histogram(col(1),buckets) places {2,4,1,0} in col(3). Two of the values in column 1 fall under 25, four fall under 50, one under 75, and no values fall outside the range.

if

The if function either selects one of two values based on a specified condition, or proceeds along a series of calculations bases on a specified condition.

Syntax

```
if(condition,true value,false value)
```

The *true value* and *false value* arguments can be any scalar or range. For a true *condition*, the true value is returned; for a false condition, the false value is returned.

If the condition argument is scalar, then the entire true value or false value argument is returned.

If the condition argument contains a range, the result is a new range. For each true entry in the condition range, the corresponding entry in the true value argument is returned. For a false entry in the condition range, the corresponding entry in false value is returned.

If the false value is omitted and the condition entry is false, the corresponding entry in the true value range is omitted. This can be used to conditionally extract data from a range.

Example 1

The operation `col(2) = if(col(1)< 75,”FAIL","PASS")` reads in the values from column 1, and places the word “FAIL” in column 2 if the column 1 value is less than 75, and the word “PASS” if the value is 75 or greater.

Example 2

For the operation `y = if(x < 2 or x > 4,99,x)`, an x value less than 2 or greater than 4 returns a y value of 99, and all other x values return a y value equal to the corresponding x value.

If you set `x = {1,2,3,4,5}`, then y is returned as {99,2,3,4,99}. The condition was true for the first and last x range entries, so 99 was returned. The condition was false for x = 2, 3, and 4, so the x value was returned for the second, third, and fourth x values.

if...then...else

The if...then...else function proceeds along one of two possible series of calculations based on a specified condition.

Syntax

```
if condition then
 statement
 statement...
else
 statement
 statement...
end if
```

To use the if...then...else construct, follow the if *condition* then statement by one or more transform equation statements, then specify the else statement(s). When an if...then...else statement is encountered, all functions within the statement are evaluated separately from the rest of the transform.

Important: You must separate if, then, and all condition statement operators, variables, and values with spaces.

Inside if...then...else constructs, you can:

- Type more than one equation on a line
- Indent equations.
- Nest additional if constructs.

Note that these conditions are allowed only within if...else statements. You cannot redefine variable names within an if...then...else construct.

Example

The operations:

```
 i = cell(1,1)
j = cell(1,2
If i < 1 and j > 1 then x = col(3)
else x = col(4)
```

```
end if
```

sets x equal to column 3 if i is less than 1 and j is greater than 1; otherwise, x is equal to column 4.

imaginary (img)

The imaginary function strips the imaginary values out of a range of complex numbers.

Syntax

```
img(block)
```

The range is made up of complex numbers.

Example

If x = {{1,2,3,4,5,6,7,8,9,10}, {0,0,0,....0,0}}, the operation `img(x)` returns {0,0,0,0,0,0,0,0,0,0}.

If x = {{1.0,-0.75, 3.1}, {1.2,2.1,-1.1}}, the operation `img(x)` returns {1.2,2.1,-1.1}.

implicit

In applications, a function is often defined implicitly by an equation involving a dependent variable and one or more independent variables. Use the *implicit* function to solve the equation for the dependent variable when a value for each independent variable has been specified.

Syntax

```
rv = implicit(expr, a, b, indvar, maxroots, firstroot)
```

The expression *expr* contains the dependent variable and the independent variables. The equation is defined by setting this expression equal to zero. The expression must be a user-defined function. For example, the user-defined function might be:

```
k(u,v) = u^2 - 3*u*v + v^2
```

and the equation being solved becomes k(u,v) = 0. In this case, you would set the first argument in the *implicit* function to k(u,v).

It is assumed that the dependent variable (the solution variable) is always the first variable in the argument list of the user-defined function. In the above example, this variable is *u*.

The argument list can have any number of independent variables, such as:

```
r(u,v,w) = (u+v*w)*exp(u*w)
```

In this case, *u* is the dependent variable with *v* and *w* as the independent variables.

The user-defined function may contain other variables than those in its argument list. These parameters are assumed to have assigned values when the function is defined. It is important that each *parameter* be a scalar quantity defined by a single numeric value. If any parameter is a range, then only the first value in that range is used.

a and *b* are the left and right endpoints, respectively, of the interval over which the solution search for the dependent variable takes place.

indvar is an expression or symbol that defines the values of the independent variables that are used to solve the equation. It is usually defined by some range. For example, suppose we have transform:

```
x=col(1)
f=implicit(k(u,v),-10,10,x)
```

Then this transform takes a value from column 1, assigns it to the variable *v*, and solves the equation k(u,v)=0 for the variable *u*. This is then repeated until all values in column 1 have been used. The results are then assigned to the variable *f*.

If more than one independent variable is listed in the user-defined function, then the values of the independent variables are concatenated in the order they appear in the function and the result is assigned to *indvar*. For example, in the transform below we have the implicit function used with two independent variables:

```
x=col(1)
y=col(2)
k(u,v,w)=u^2+v^2+w
col(4)=implicit(k(u,v,w),-10,10,{x,y})
```

In this example, a value from column 1 is assigned to the variable *v* and a value for column 2, in the same row, is assigned to variable *w*. The equation k(u,v,w) is then solved for the variable *u*. This process is repeated until there are no more entries in columns 1 and 2.

It is important to note that when a range of data is provided to the argument *indvar*, then that data is divided equally among the independent variables in the equation. The data will be partitioned this way, beginning with the first independent variable, and the implicit function will ignore the remainder. For example, if *indvar* is provided a range of 14 values and there are 3 independent variables in the problem, then the values 1-4 will be assigned to the first variable, values 5-8 will be assigned to the second variable, and values 9-12 will be assigned to the third variable. The last two values will be ignored.

maxroots is an optional argument, representing the maximum number of dependent variable values to compute for each specified set of values of the independent variables. The default value is 1.

firstroot is an optional argument that indicates how the function returns the solutions. The default value is zero.

The Return Value, or *rv*, is the list or range of all of the solutions that were found. The number of values returned will always be equal to *maxroots* for each set of independent variable values. If fewer *roots* than maxroots are found, then the remaining values returned will be missing values. The reason for inserting the missing values is so the output of the different functions can be distinguished.

Helpful Tips

- Increasing the value of *maxroots* increases the chances of finding all of the solutions of the equation in the prescribed interval. It also increases the time required to complete the processing of the implicit function.
- When searching for multiple solutions to an equation, the *implicit* function partitions the interval that you specify into *maxroots* equally-spaced subintervals. It then searches each subinterval for exactly one solution. As a consequence, the implicit function may return fewer than *maxroots* solutions, even though the equation actually has *maxroots* or more solutions in the supplied interval. Ideally, to find all of the solutions to the equation over the interval from *a* to *b*, set maxroots to a value greater than (b -a)/delta, where delta estimates the closest distance between any solutions.
- The output of the implicit function is always sorted to give the roots in ascending order for each function in *expr*.

Example 1

This first example finds a solution of the equation a*u^b/(c^b+u^b) - v =0 for each value of v in the sampled data from the interval from 0 to 10. The results are written to column 1.

```
a=95
b=3
c=1.5
x=data(0,10,.025)
k(u,v)=a*u^b/(c^b+u^b) - v
col(1)=implicit(k(u,v),0,10,x)
```

Example 2

Graphing an implicit equation with two variables can be difficult. The obvious way is to select several values of one variable, which will be referred to as the independent variable, and solve the equation for the remaining variable, which will be the dependent variable. In order to get the complete graph of the equation over a given range of values for the independent variable, we need to obtain all of the solutions of the equation for the dependent variable.

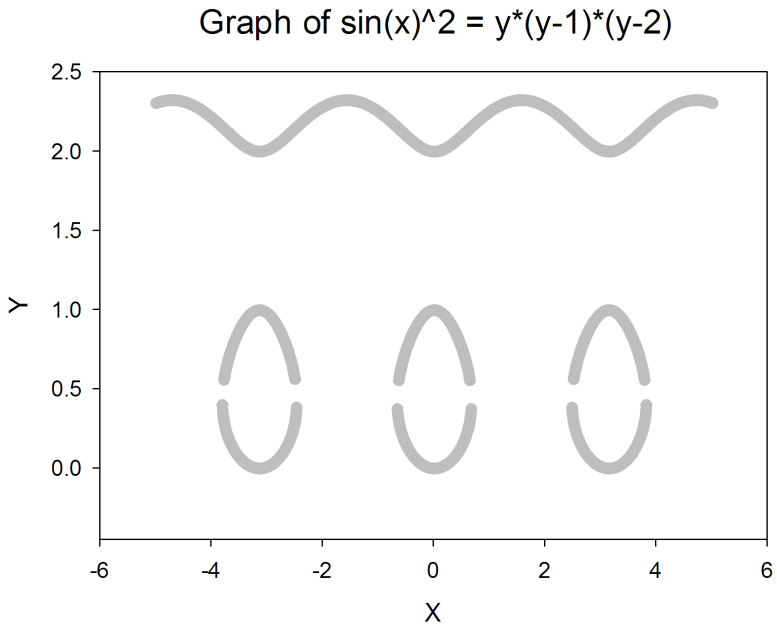
Suppose we wish to graph the equation sin(x)^2 = y*(y-1)*(y-2) for values of *x* between -5 and 5. It is clear that the y values for this equation must be greater than 0, otherwise sin(x)^2 would be negative, which is impossible. Also, any *y* value must be less than 3, otherwise the right side of the equation would be greater than 1, which is the largest value of sin(x)^2. Thus, we will choose our search interval for y between 0 and 3. Looking at the equation, we see that for each value of x there are at most 3 values of y since the right side of the equation is a cubic polynomial. Knowing this, we could set *maxroots* equal to 3. However, from the discussion in the remarks above, since we don't know how close the solutions are for a given x value, we will set this value higher to *maxroots* = 10.

The transform below generates the data that will be used to obtain the graph.

```
x=data(-5,5,.005)
k(u,v)=u*(u-1)*(u-2)-sin(v)^2
col(2)=implicit(k(u,v),0,3,x,10)
for i=1 to size(x) do
    for j = 1 to 10 do
       cell(1,10*(i-1) + j) = x[i]
    end for
end for
```

The first line samples several values of the independent variable over the interval from -5 to 5. The second line defines the expression that will be set to zero to give us the equation. The call to the implicit function in the third line contains our information for the search interval and maxroots. Because of how the implicit function arranges the output, with all solutions for each x-value displayed consecutively, the remaining lines of the transform arrange our x-data in the worksheet by repeating a value for the 10 corresponding solutions. Note that there will be many missing values in the output of the implicit function since we know that a maximum of three values is all we expect. When graphing, these missing values will simply be ignored. Before running the transform, make sure that radians is selected as the angular units.

After running the transform, create a simple scatter plot with columns 1 and 2 selected for the XY Pair data format. The resulting graph is:



The three nearly closed curves below the undulating curve should indeed be closed, but more sampled points are needed.

Example 3 (Implicit Curve Fitting)

This example shows the equation text for a curve fit in which the fit model is defined implicitly. In this particular example, the data in the table is fit by an ellipse that is defined implicitly.

When using the *implicit* function in curve fitting, you can set the value of *maxroots* to any value which is sufficient to locate one solution. However, if *maxroots* is greater than 1, you must set the value of *firstroot* to 1 (or follow the call of the implicit function with a method of selecting one solution), for otherwise the returned values cannot be properly matched to the independent variable values.

-2.0000	0.1000	[Variables]
-1.5000	0.3000	x=col(1)
-1.0000	0.4000	y=col(2)
-0.5000	0.6000	[Parameters]
0.0000	0.5000	a = .01
0.5000	0.6000	b = .01
1.0000	0.5000	[Equation]
1.5000	0.2500	k(u,v)=a*v^2+b*u^2-1
2.0000	0.1000	f=implicit(k(u,v),0,10,x) fit f to y [Constraints] a>0 b>0 [Options] tolerance=1e-010 stepsize=1 iterations=200

int

The int function returns a number or range of numbers equal to the largest integer less than or equal to each corresponding number in the specified range. All numbers are rounded down to the nearest integer.

Syntax

```
int(numbers)
```

The *numbers* argument can be a scalar or range of numbers. Any missing value or text string contained within a range is ignored and returned as the string or missing value.

Example

The operation `int({.9,1.2,2.2,-3.8})` returns a range of {0.0,1.0,2.0,-4.0}.

interpolate

The interpolate function performs linear interpolation on a set of X,Y pairs defined by an x range and a y range. The function returns a range of interpolated y values from a range of values between the minimum and maximum of the x range.

Syntax

```
interpolate(x range,y range,range)
```

Values in the *x range* argument must be strictly increasing or strictly decreasing.

The *range* argument must be a single range (indicated with the {} brackets) or a worksheet column. Missing values and text strings are not allowed in the *x range* and *y range*. Text strings in range are replaced by missing values.

Extrapolation is not possible; missing value symbols are returned for range argument values less than the lowest *x* range value or greater than the highest *x* range value.

Example

For x = {0,1,2}, y = {0,1,4}, and range = data(0,2,.5) (this data operation returns numbers from 0 to 2 at increments of 0.5), the operation `col(1) = interpolate(x,y,range)` places the range {0.0,0.5,1.0,2.5,4.0} into column 1.

If *range* had included values outside the range for x, missing values would have been returned for those out-of-range values.

inv

The inv function generates the inverse matrix of an invertible square matrix provided as a block.

Syntax

```
inv(block)
```

The *block* argument is a block of numbers with real values in the form of a square matrix. The number of rows must equal the number of columns. The function returns a block of numbers with real values in the form of the inverse of the square matrix provided.

Example

For the matrix:

1.00	3.00	4.00
2.00	1.00	3.00
3.00	4.00	2.00

in block(2,3,4,5) the operation `block(2,7)=inv(block(2,3,4,5))` generates the inverse matrix:

-0.40	0.40	0.20
0.20	-0.40	0.20
0.20	0.20	-0.20

in block (2,7,4,9).

invcpx

This function takes the reciprocal of a range of complex numbers.

Syntax

```
invcp(block)
```

The input and output are blocks of complex numbers. The invcpx function returns the range 1/c for each complex number in the input block.

Example

If x = complex ({3,0,1}, {0,1,1}), the operation `invcpx(x)` returns {{0.33333, 0.0, 0.5}, {0.0,-1.0,-0.5}}.

invfft

The inverse fft function (invfft) takes the inverse Fast Fourier Transform (fft) of the data produced by the fft to restore the data to its new filtered form.

Syntax

```
invfft(block)
```

The parameter is a complex block of spectral numbers with the real values in the first column and the imaginary values in the second column. This data is usually generated from the fft function. The invfft function works on data sizes of size 2^n numbers. If your data set is not 2^n in length, the invfft function pads 0 at the beginning and end of the data range to make the length 2^n.

The function returns a complex block of numbers.

Example

If x = {{1,2,3,...,9,10}, {0,0,0,...,0,0}}, the operation `invfft(fft(x))` returns {{0,0,0,1,2,3,...,9,10,0,0,0}, {0,0,0,...0,0}.

linsys

The linsys function solves systems of linear equations and multiple linear regression problems.

Syntax

```
linsys (block, range, extended info
```

The *block* argument provides a matrix whose interpretation depends on the type of problem being solved. If solving a system of linear equations, this argument provides the coefficient matrix for the system, where each variable in the system corresponds to a column of the matrix and where each row gives the coefficients in one of the (scalar) equations of the system. If solving a multiple linear regression problem, this argument provides the matrix of sampled values of the independent variables in the model, often called the design matrix, and may also include a column of ones if the regression model contains a constant term.

The *range* argument provides a range of values whose interpretation, like the block argument, depends on the type of problem being solved. If solving a system of linear equations, this argument provides the constants on the right-hand sides of the equations. If solving a multiple linear regression problem, this argument provides the range of observations that are being fitted. The *extended info* argument is an optional value. If this argument is not provided or has a value equal to 1, then the function returns a block with two columns:

- The first column is a range of values that corresponds to the solution of the problem. The size of this range equals number of columns in the block argument. If solving a system of linear equations, the range contains values for the variables that solve the system. If solving a multiple linear regression problem, the range contains the values of the regression parameters that give the best linear model for the data.
- The second column contains two numbers. The first number is the *Euclidean distance* from the *range* argument to vector obtained by multiplying the matrix represented by the *block* argument by the solution vector in the first column. The second number is a numerical estimate of the *rank* of the matrix represented by the *block* argument. The rank is an integer and gives the number of linearly independent columns in the matrix. The rank cannot exceed the minimum of the number of rows and columns in the matrix.

  If solving a system of linear equations, a distance value very small, relative to the data in the problem, indicates the range of values in the first column provides a solution. In this case, the rank value is used to determine if this solution is unique. If the rank value equals the number of columns of the coefficient matrix, then the solution is unique; otherwise, there are infinitely many solutions and the solution that is given in the first column is the solution of smallest length, or smallest Euclidean norm. If the distance value is not small, this indicates the system is inconsistent (no solution) and the solution vector in the first column actually represent values of the system variables so that the left sides of equations “come closest” to the constants on the right-hand sides of the

equations. If a system is inconsistent, then it is necessarily true that the rank is less than the number of rows in the coefficient matrix.

If solving a multiple linear regression problem, the distance value equals the square root of the residual sum of squares and measures the distance between the observations and the predicted values of the regression model. A distance very close to zero indicates a perfect fit. A rank value less than the number of columns of the design matrix indicates the solution in the first column is not unique. In this case, the solution obtained in the first column is the minimum norm solution, meaning the solution whose *Euclidean norm* is the smallest.

If the *extended info* value is 0, then only the range for the solution is returned, without the values for Euclidean distance or rank.

Example 1

For the system of equations:

$x + 2y + z = 1$

$2x - 3y - z = -4$

$x - y + 3z = 1$

enter the coefficient matrix:

1	2	1
2	3	1
1	1	3

in block (1,1,3,3) and the right-hand sides of the equations

1
-4
1

in col (4,1,3). The operation

```
block (5,1) = linsys (block (1,1,3,3), col (4,1,3))
```

returns the block

-0.8261	4.4409e-16
0.5217	3.0000
0.7826	

in block (5,1,6,3).

The first column gives the solution vector for the system in the same order in which the variables appear in the equations (x = -0.8261, y = 0.5217, z = 0.7826). The rank value 3.0000 equals the number of columns and the number of rows of the coefficient matrix, indicating that a solution exists and is unique. The distance value 4.4409e-16 is small compared with the order of magnitude of the data in the problem, verifying our solution to the problem.

Example 2

For the system of equations:

$2x - y + 3z = 7$
$x - 3y - z = 11$
$x + 2z = 2$

Enter the coefficient matrix

2	1	3
1	3	1
1	0	2

in block (1,1,3,3) and the right-hand sides of the equations

7
11
2

in col (4,1,3). The operation

```
block (5,1) = linsys (block (1,1,3,3), col (4,1,3))
```

returns

1.6667	9.5763e-15
-3.1667	2.0000
0.1667	

in block (5,1,6,3).

The first column gives the solution vector for the system in the same order in which the variables appear in the equations (x = 1.6667, y = -3.1667, z = 0.1667). The rank value 2.0000 shows that this solution is not unique. The solution that is given is the solution of smallest Euclidean norm. Another solution is x = 2.0, y = -3.0, z = 0.0. The distance value 9.5763e-15 is small compared with the order of magnitude of the data in the problem, verifying our solution to the problem.

Example 3

Suppose we want to fit the model $y = ax + by + cz$ to the observations

6.20
26.90
7.60
-18.60
14.60
-23.40
19.40

in col (1,1,7) corresponding to the independent variable values

26-x	27-y	28-z

1.00 1.00 1.00
1.00 2.00 2.00
1.00 3.00 3.00
2.00 2.00 3.00
2.00 1.00 1.00
3.00 2.00 6.00
3.00 3.00 -1.00

in block (2,1,4,7). The operation

```
block (5,1) = linsys (block (2,1,4,7), col (1,1,7))
```

returns

-3.2601 32.1904
9.9303 3.0000
-5.2716

in block (5,1,6,3).

The first column gives the solution vector for the system in the same order in which the parameters appear in the equations (a = -3.2601, b = 9.9303, c = -5.2716). The rank value 3.0000 shows that this solution is unique. The distance value 32.1904 equals the Euclidean distance from the observation vector to the vector of predicted values which is obtained by multiplying the design matrix in block (2,1,4,7) by the solution vector in the first column of the results.

ln

The ln function returns a value or range of values consisting of the natural logarithm (*base e)* of each number in the specified range.

Syntax

```
ln(numbers)
```

The numbers argument can be a scalar or range of numbers. Any missing value or text string contained within a range is ignored and returned as the string or missing value.

For `ln(x)`:

`x < 0` returns an error message, and

`x = 0` returns -∞

The largest value allowed is approximately $x < 10^{309}$.

Example

The operation `ln(2.71828)` returns a value ≈ 1.0.

log

The log function returns a value or range of values consisting of the base 10 logarithm of each number in the specified range.

Syntax

```
log(numbers)
```

The *numbers* argument can be a scalar or range of numbers. Any missing value or text string contained within a range is ignored and returned as the string or missing value.

For `log(x)`:

`x < 0` returns an error message,

`x = 0` returns -∞

The largest value allowed is approximately $x < 10^{309}$.

Example

The operation `log(100)` returns a value of 2.

logisden

This function is the logistic distribution's probability density function. It returns the value of the slope of the cumulative distribution function at the specified argument value.

Syntax

```
logisden(x,a,b)
```

The x argument represents the independent variable and can either be a scalar or a range of numbers. If x is a range, then it must be defined by either using braces { } or by specifying a worksheet column. Any value for x must be real. The a argument is any real number and is the location parameter, equal to the location of the density function's peak. The b argument is any positive number and is the scale parameter.

Example

The density function can be used to estimate the probability that the values of a logistic random variable X lie in a small interval. If X has a location parameter equal to 3 and a shape parameter equal to 4, then to estimate the probability that the values of X lie between 2 and 2.1, multiply the density of X at 2 by the length of the interval .1:

```
logisden(2,3,1) * .1 = .019661
```

logisdist

This function is the cumulative logistic distribution function. It returns the probability that a logistic random variable is less than a specified independent variable value.

The distribution of a logistic random variable is similar in shape to the normal distribution, but with wider tails (higher kurtosis).

Syntax

```
logisdist(x,a,b)
```

The x argument represents the independent variable and can either be a scalar or a range of numbers. If x is a range, then it must be defined by either using braces { } or by specifying a worksheet column. Any value for x must be real. The a argument is any real number and is the location parameter, equal to the location of the density function's peak. The b argument is any positive number and is the scale parameter.

Example

Suppose a logistic random variable X has location parameter equal to 3 and shape parameter equal to 1. To compute the probability that the values of X exceed 2, we calculate:

```
P( X > 2 ) = 1 – P( X < 2 ) = 1 – logisdist(2,3,1) = .73106
```

logisinv

This function is the inverse cumulative logistic distribution function. The probability that a logistic random variable is less than the return value is equal to the argument you specify.

Syntax

```
logisinv(x,a,b)
```

The x argument can either be a scalar or a range of numbers. If x is a range, then it must be defined by either using braces { } or by specifying a worksheet column. Any scalar value for x represents a probability and so must be between 0 and 1. The a argument is any real number and is the location parameter, equal to the location of the density function's peak. The b argument is any positive number and is the scale parameter.

Example

Suppose a logistic random variable X has location parameter equal to 3 and shape parameter equal to 1. To calculate the tail of this distribution whose probability is .05, we need to find a number x such that P(X > x) = .05. This is the same as finding x such that P(X < x) = .95. Therefore, we calculate:

```
logisinv(.95, 3, 1) = 5.94444
```

loglogisden

This function is the log-logistic distribution's probability density function. It returns the value of the slope of the cumulative distribution function at the specified argument value.

Syntax

```
loglogisden(x,a,b)
```

The x argument represents the independent variable and can either be a scalar or a range of numbers. If x is a range, then it must be defined by either using braces { } or by specifying a worksheet column. Any value for x must be real. The a argument is any real number and is the location parameter. The b argument is any positive number and is the scale parameter.

Example

The density function can be used to estimate the probability that the values of a log-logistic random variable X lie in a small interval. If X has location parameter equal to 1 and shape parameter equal to 1, then to estimate the probability that the values of X lie between 2 and 2.1, multiply the density of X at 2 by the length of the interval .1:

```
loglogisden(2,1,1) * .1 = .012210
```

loglogisdist

This function is the cumulative log-logistic distribution function. It returns the probability that a log-logistic random variable is less than a specified independent variable value.

The log-logistic distribution function gives the distribution of the random variable Y = exp(X), where X has a logistic distribution.

Syntax

```
loglogisdist(x,a,b)
```

The *x* argument represents the independent variable and can either be a scalar or a range of numbers. If *x* is a range, then it must be defined by either using braces { } or by specifying a worksheet column. Any value for *x* must be real. The *a* argument is any real number and is the location parameter. The *b* argument is any positive number and is the scale parameter.

Example

Suppose a log-logistic random variable X has location parameter equal to 1 and shape parameter equal to 1. To compute the probability that the values of X exceed 2, we calculate:

```
P( X > 2 ) = 1 – P( X < 2 ) = 1 – loglogisdist(2,1,1) = .57612
```

loglogisinv

This function is the inverse cumulative log-logistic distribution function. The probability that a log-logistic random variable is less than the return value is equal to the argument you specify.

Syntax

```
loglogisinv(x,a,b)
```

The *x* argument can either be a scalar or a range of numbers. If *x* is a range, then it must be defined by either using braces { } or by specifying a worksheet column. Any scalar value for *x* represents a probability and so must be between 0 and 1. The *a* argument is any real number and is the location parameter. The *b* argument is any positive number and is the scale parameter.

Example

Suppose a log-logistic random variable X has location parameter equal to 1 and scale parameter equal to .5. To calculate the tail of this distribution whose probability is .05, we need to find a number x such that P(X > x) = .05. This is the same as finding x such that P(X < x) = .95. Therefore, we calculate:

```
loglogisinv(.95, 1, .5) = 11.84872
```

lognormden

This function is the log-normal distribution's probability density function. It returns the value of the slope of the cumulative distribution function at the specified argument value.

Syntax

```
lognormden(x,a,b)
```

The x argument represents the independent variable and can either be a scalar or a range of numbers. If x is a range, then it must be defined by either using braces { } or by specifying a worksheet column. Any value for x must be real. The a argument is any real number and is the location parameter. The b argument is any positive number and is the scale parameter.

Example

The density function can be used to estimate the probability that the values of a log-normal random variable X lie in a small interval. If X has location parameter equal to 1 and shape parameter equal to 1, then to estimate the probability that the values of X lie between 2 and 2.1, multiply the density of X at 2 by the length of the interval .1:

```
lognormden(2,1,1) * .1 = .019030
```

lognormdist

This function is the cumulative log-normal distribution function. It returns the probability that a log-normal random variable is less than a specified independent variable value.

The log-normal distribution function gives the distribution of the random variable Y = exp(X), where X has a normal distribution.

Syntax

```
lognormdist(x,a,b)
```

The x argument represents the independent variable and can either be a scalar or a range of numbers. If x is a range, then it must be defined by either using braces { } or by specifying a worksheet column. Any value for x must be real.

The a argument is any real number and is the location parameter. The b argument is any positive number and is the scale parameter.

Example

Suppose a log-normal random variable X has location parameter equal to 1 and shape parameter equal to 1. To compute the probability that the values of X exceed 2, we calculate:

```
P( X > 2 ) = 1 – P( X < 2 ) = 1 – lognormdist(2,1,1) = .62052
```

lognorminv

This function is the inverse cumulative log-normal distribution function. The probability that a log-normal random variable is less than the return value is equal to the argument you specify.

Syntax

```
lognorminv(x,a,b)
```

The x argument can either be a scalar or a range of numbers. If x is a range, then it must be defined by either using braces { } or by specifying a worksheet column. Any scalar value for x represents a probability and so must be between 0 and 1. The a argument is any real number and is the location parameter. The b argument is any positive number and is the scale parameter.

Example

Suppose a log-normal random variable X has location parameter equal to 1 and scale parameter equal to .5. To calculate the tail of this distribution whose probability is .05, we need to find a number x such that P(X > x) = .05. This is the same as finding x such that P(X < x) = .95. Therefore, we calculate:

```
lognorminv(.95, 1, .5) = 6.18686
```

lookup

The lookup function compares values with a specified table of boundaries and returns either a corresponding index from a one-dimensional table, or a corresponding value from a two-dimensional table.

Syntax

```
lookup(numbers,x table,y table)
```

The *numbers* argument is the range of values looked up in the specified *x table*. The x table argument consists of the upper bounds (inclusive) of the *x* intervals within the table and must be ascending in value. The lower bounds are the values of the previous numbers in the table (-∞ for the first interval).

You must specify numbers and an *x table*. If only the numbers and x table arguments are specified, the lookup function returns an index number corresponding to the x table interval; the interval from -∞ to the first boundary corresponds to an index of 1, the second to 2, etcetera.

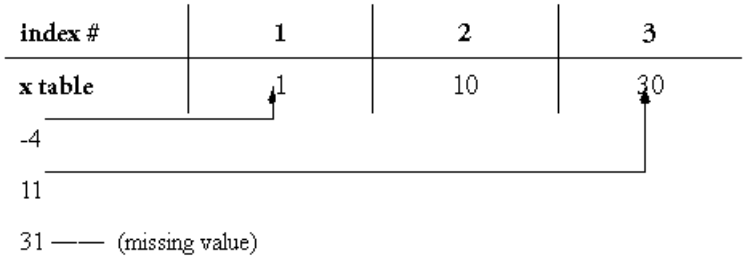
If a number value is larger than the last entry in *x table*, lookup will return a missing value as the index. You can avoid missing value results by specifying 1/0 (infinity) as the last value in *x table*.

The optional *y table* argument is used to assign *y* values to the x index numbers. The y table argument must be the same size as the *x* table argument, but the elements do not need to be in any particular order. If y table is specified, lookup returns the y table value corresponding to the x table index value, i.e., the first y table value for an index of 1, the second y table value for an index of 2, etc.

Note: The x table and y table ranges correspond to what is normally called a “lookup table.”

Example 1

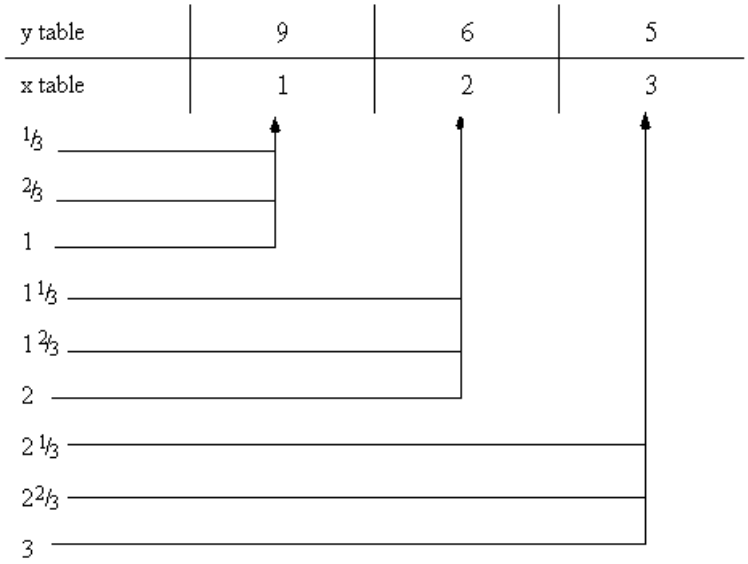
For n={-4,11,31} and x={1,10,30}, `col(1)=lookup(n,x)` places the index values of 1, 3, and -- (missing value) in column 1.



-4 falls beneath 1, or the first x boundary; 11 falls beyond 10 but below 30, and 31 lies beyond 30.

Example 2

To generate triplet values for the range {9,6,5}, you can use the expression lookup(data(1/3,3,1/3),data(1,3),{9,6,5}) to return {9,9,9,6,6,6,5,5,5}. This looks up the numbers 1/3, 2/3, 1, 1 1/3, 1 2/3, 2, 2 1/3, 2 2/3, and 3 using x table boundaries 1, 2, and 3 and corresponding y table values 9, 6, and 5.



lowess

The lowess function returns smoothed y values as a range from the ranges of x and y variables provided, using a user-defined smoothing factor. "Lowess" means *locally weighted regression*. Each point along the smooth curve is obtained from a regression of data points close to the curve point with the closest points more heavily weighted.

Syntax

```
lowess(x range, y range, f )
```

The *x range* argument specifies the x variable, and the *y range* argument specifies the y variable. Any missing value or text string contained within one of the ranges is ignored and will not be treated as a data point. *x range* and *y range* must be the same size, and the number of valid data points must be greater than or equal to 3.

The *f* argument defines the amount of Lowess smoothing, and corresponds to the fraction of data points used for each regression. *f* must be greater than or equal to 0 and less than or equal to 1. $0 \le f \le 1$. Note that unlike lowpass, lowess requires an *f* argument.

Example

For x = {1,2,3,4}, y={0.13, 0.17, 0.50, 0.60}, the operation

```
col(1)=lowess(x,y,1)
```

places the smoothed y data 0.10, 0.25, 0.43, 0.63 into column 1.

lowpass

The lowpass function returns smoothed y values from ranges of x and y variables, using an optional user-defined smoothing factor that uses FFT and IFFT.

Syntax

```
lowpass(x range, y range, f )
```

The *x range* argument specifies the x variable, and the *y range* argument specifies the y variable. Any missing value or text string contained within one of the ranges is ignored and will not be treated as a data point. *x range* and *y range* must be the same size, and the number of valid data points must be greater than or equal to 3.

The optional *f* argument defines whether FFT and IFFT are used. f must be greater than or equal to 0 and less than or equal to 100 . If f is omitted, no Fourier transformation is used.

Note: *lowpass* is especially designed to perform smoothing on waveform functions as a part of nonlinear regression.

Example

For x = {0,1,2}, y={0,1,4}, the operation

```
col(1)=lowpass(x,y,88)
```

places the newly smoothed data 0.25, 1.50, 2.25 into column 1.

makeblock

The makeblock function returns a range or block of data having a specified size using a range of data that you supply.

Syntax

```
makeblock(number of columns, number of rows, scalar or range)
```

The first and second arguments are numbers that specify the size of the output block, or output range if the first argument equals 1. The third argument provides the data used to fill the entries in the output.

If the third argument is a scalar, either a number or a string, then all entries of the output block or range are set to that scalar. If the third argument is a range, then its entries are taken in order to fill the output block one column at a time, starting each column at the first row. If there are more entries in the input range than the output block can use, then only the entries needed to fill the output block will be used. If there are fewer entries in the input range than the output block requires, then the entries of the range are recycled as many times as necessary to fill the entries of the output block.

Example 1

To delete the contents of column 1, use `col(1) = makeblock(1, size(col(1)),"")`, where the last argument is a pair of double quotes or an empty string. To delete the contents of rows 3 through 6 of column 1, use `col(1,3,6) = makeblock(1,4,"")`.

Example 2

To create a 3 by 3 block of 1s in the worksheet whose upper left cell is in row 1 of column 1, then use `block(1,1)=makeblock(3,3,1).`

Example 3

The output of the transform `block(2,1)=makeblock(2,3,{3,2,4,4,6,7})` will appear in the worksheet as:

3 4

2 6

4 7

The top-left cell is in column 2, row 1.

Example 4

Suppose you have blank cells distributed with the data in some column, say column 1. To filter out the blank cells from column 1 in-place, you could use the following transform

```
a=if(col(1) <> "", col(1))
col(1)= makeblock(1,size(col(1)),"")
col(1)=a
```

This transform puts the filtered contents into memory, deletes the contents of column 1, and then puts the filtered contents into column 1.

max

The max function returns the largest number found in the range specified.

Syntax

```
max(range)
```

The *range* argument must be a single range (indicated with the { } brackets) or a worksheet column. Any missing value or text string contained within a range is ignored.

Example

For `x = {7,4,-4,5}`, the operation `max(x)` returns a value of 7, and the operation `min(x)` returns a value of -4.

mean

The mean function returns the average of the range specified. Use this function to calculate column averages (as opposed to using the avg function to calculate row averages).

The mean function calculates the arithmetic mean, defined as:

$$x = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Syntax

```
mean(range)
```

The *range* argument must be a single range (indicated with the { } brackets) or a worksheet column. Any missing value or text string contained within a range is ignored.

Example

The operation `mean({1,2,3,4})` returns a value of 2.5.

median

The median function returns the median of the range specified. The median is a number that is both less than or equal to half and greater than or equal to half of the values in the data set.

The median of a set with an odd number of elements is simply the middle value when the elements are sorted by value. For an even number of elements, the median returns the mean of the two middle values when the elements are sorted by value.

Syntax

```
median(range)
```

The *range* argument must be a single range (indicated with the { } brackets) or a worksheet column. Any missing value or text string contained within a range is ignored.

Example

The operation median({1,2,3,5}) returns a value of 2.5.

min

The min function returns the smallest number in the range specified.

Syntax

```
min(range)
```

The *range* argument must be a single range (indicated with the { } brackets) or a worksheet column. Any missing value or text string contained within a range is ignored.

Example

For x = {7,4,-4,5}, the operation `max(x)` returns a value of 7, and the operation `min(x)` returns a value of -4.

missing

The missing function returns a value or range of values equal to the number of missing values and text strings in the specified range.

Syntax

```
missing(range)
```

The range argument must be a single range (indicated with the { } brackets) or a worksheet column.

mod

The mod function returns the modulus (the remainder from division) for corresponding numbers in numerator and divisor arguments. This is the real (not integral) modulus, so both ranges may be nonintegral values.

Syntax

```
mod(numerator,divisor)
```

The *numerator* and *divisor* arguments can be scalars or ranges. Any missing value or text string contained within a range is returned as the string or missing value.

For any divisor ≠ 0, the mod function returns the remainder of $\frac{numerator}{divisor}$.

For `mod(x,0)` , that is, for *divisor* = 0,

`x > 0` returns +∞

`x = 0` returns + ∞

`x < 0` returns - ∞

Example

The operation `mod({4,5,4,5},{2,2,3,3})` returns the range {0,1,1,2}. These are the remainders for 4÷2, 5÷2, 4÷3, and 5÷3.

mulcpx

The mulcpx function multiplies two blocks of complex numbers together.

Syntax

```
mulcpx(block, block)
```

Both input blocks should be the same length. The mulcpx function returns a block that contains the complex multiplication of the two ranges.

Example

If u = {{1,1,0},{0,1,1}}, the operation `mulcpx(u,u)` returns {{1,0,-1}, {0,2,0}}.

normden

This function is the normal (or Gaussian) probability density function. The graph of this function is the familiar "bell curve". It returns the value of the slope of the cumulative distribution function at the specified argument value.

Syntax

```
normden(x,m,s)
```

The *x* argument represents the independent variable and can either be a scalar or a range of numbers. If *x* is a range, then it must be defined by either using braces { } or by specifying a worksheet column. The *m* argument can be any number and equals the mean of the distribution. The *s* argument can be any positive number and equals the standard deviation of the distribution.

Example

The density function can be used to estimate the probability that the values of a normally distributed random variable X lie in a small interval. If X has mean 0 and standard deviation 1, then to estimate the probability that the values of X lie between .5 and .6, multiply the density of X at .5 by the length of the interval .1:

```
normden(.5,0,1) * .1 = .03521
```

normdist

This function is the cumulative normal (or Gaussian) distribution function. It returns the probability that a normal random variable is less than a specified independent variable value.

Syntax

```
normdist(x,m,s)
```

The *x* argument represents the independent variable and can either be a scalar or a range of numbers. If *x* is a range, then it must be defined by either using braces { } or by specifying a worksheet column. The *m* argument can be any number and equals the mean of the distribution. The *s* argument can be any positive number and equals the standard deviation of the distribution.

A normal distribution is called *standard* if the mean is 0 and the standard deviation is 1.

Example

Suppose a random variable X is normally distributed withmean .5 and standard deviation 2. Then to compute the probability that its values lie between -1 and 1, we calculate:

```
P(-1 < X < 1) = P(X < 1) - P(X < -1)       = normdist(1,.5,2) -
 normdist(-1,.5,2)          = .59871 - .22663 = .37208
```

norminv

This function is the inverse cumulative normal (or Gaussian) distribution function. The probability that a normally distributed random variable is less than the return value is equal to the argument you specify.

Syntax

```
norminv(x,m,s)
```

The *x* argument can either be a scalar or a range of numbers. If *x* is a range, then it must be defined by either using braces { } or by specifying a worksheet column. Any scalar value for *x* represents a probability and so must be between 0 and 1. The *m* argument can be any number and equals the mean of the distribution. The *s* argument can be any positive number and equals the standard deviation of the distribution.

Example

Suppose a random variable X is normally distributed with mean .5 and standard deviation 2. To compute the .25 quartile of X, we calculate:

```
norminv(.25,.5,2) = -.84898
```

nth

The nth function returns a sampling of a provided range, with the frequency indicated by a scalar number. The result always begins with the first entry in the specified range.

Syntax

```
nth(range,increment)
```

The *range* argument is either a specified range (indicated with the {} brackets) or a worksheet column. The *increment* argument must be a positive integer.

Example

The operation `col(1)=nth({1,2,3,4,5,6,7,8,9,10},3)` places the range {1,4,7,10} in column 1. Every third value of the range is returned, beginning with 1.

partdist

The partdist function returns a range representing the distance from the first X,Y pair to each other successive pair. The line segment X,Y pairs are specified by an x range and a y range. The last value in this range is numerically the same as that returned by dist, assuming the same x and y ranges.

Syntax

```
partdist(x range,y range)
```

The *x range* argument specifies the x coordinates, and the *y range* argument specifies the y coordinates. Corresponding values in these ranges form xy pairs.

If the ranges are uneven in size, excess x or y points are ignored.

Example

For the ranges x = {0,1,1,0,0} and y = {0,0,1,1,0}, the operation `partdist(x,y)` returns a range of {0,1,2,3,4}. The X and Y coordinates provided describe a square of 1 unit x by 1 unit y.

polynomial

The polynomial function returns the results for independent variable values in polynomials. Given the coefficients, this function produces a range of y values for the corresponding x values in range.

The function takes one of two forms. The first form has two arguments, both of which are ranges. Values in the first range are the independent variable values. The second range represents the coefficients of the polynomial, with the constant coefficient listed first, and the highest order coefficient listed last.

The second form accepts two or more arguments. The first argument is a range consisting of the independent variable values. All successive arguments are scalar and represent the coefficients of a polynomial, with the constant coefficient listed first and the highest order coefficient listed last.

Syntax

```
polynomial(range,coefficents) or
polynomial(range,a0,a1,...,an)
```

The *range* argument must be a single range (indicated with the { } brackets) or a worksheet column. Text strings contained within a range are returned as a missing value.

The *coefficients* argument is a range consisting of the polynomial coefficient values, from lowest to highest. Alternately, the coefficients can be listed individually as scalars.

Example

To evaluate the polynomial $y = x^2 + x + 1$ for x values of 0, 1, and 2, type the equation `polynomial({0,1,2},1,1,1)`. Alternately, you could set x ={1,1,1}, then enter `polynomial({0,1,2},x)`. Both operations return a range of {1,3,7}.

prec

The prec function rounds a number or range of numbers to the specified number of significant digits, or places of significance. Values are rounded to the nearest integer; values of exactly 0.5 are rounded up.

Syntax

```
prec(numbers,digits)
```

The *numbers* argument can be a scalar or range of numbers. Any missing value or text string contained within a range is ignored and returned as the string or missing value.

If the *digits* argument is a scalar, all numbers in the range have the same number of places of significance.

If the *digits* argument is a range, the number of places of significance vary according to the corresponding range values. If the size of the digits range is smaller than the numbers range, the function returns missing values for all numbers with no corresponding digits.

Example

For x = {13570,3.141,.0155,999,1.92}, the operation `prec(x,2)` returns {14000,3.100,.0160,1000,1.90}.

For y = {123.5,123.5,123.5,123.5}, the operation `prec(y,{1,2,3,4})` returns {100.0, 120.0,124.0,123.5}.

prod

The prod function computes the product of two matrices where each matrix is provided as a block or a range. The product is defined as usual matrix multiplication where entries in the result are computed as the inner products of rows of the first matrix and columns of the second matrix.

Syntax

```
prod (block or range, block or range)
```

If the first argument is a range, it is interpreted as a matrix with only one row (a row vector) with its column entries equal to the entries of the range. If the second argument is a range, it is interpreted as a matrix with only one column (a column vector) with its row entries equal to the entries of the range.

To multiply two matrices, the number of columns in the first matrix argument must equal the number of rows in second matrix argument. If this condition is not satisfied, an error message is displayed and transform execution stops. The resulting product matrix will have the same number of rows as the first matrix and the same number of columns as the second matrix.

Example 1

For ranges x = {0, 3.0, 1.5, 4.0} and y = {2.0, 1.0, -5.0, 2.5}, the operation prod (x, y) returns the inner product of the ranges (or vectors) x and y:

If a = prod (x, y), then a = 0*2.0 + 3.0*1.0 + 1.5*(-5.0) + 4.0*2.5 = 5.5

Example 2

For the matrix:

Table 2:

1.00	2.00	4.00
2.00	0.00	1.00
2.00	2.00	3.00

in block (1,1,3,3), and the column vector:

1.00
1.00
2.00

in col (4), the operation

```
col (5) = prod (block (1,1,3,3), col (4))
```

returns the column vector:

11.00
4.00
10.00

in col (5).

Example 3

For the matrices:

2.00	2.00	1.00
3.00	1.00	2.00

1.00	1.00
1.00	0.00
0.00	1.00

in block (1,1,3,2) and block (4,1,5,3), respectively,

the operation

```
block (6,1) = prod (block (1,1,3,2), block (4,1,5,3))
```

returns the matrix:

4.00	1.00
4.00	5.00

in block (6,1,7,2).

Example 4

For the matrices:

1.00
2.00
-1.00

2.00	2.00	1.00

in block (1,1,1,3) and block (2,1,4,1), respectively,

the operation

```
block (5,1) = prod (block (1,1,1,3), block (2,1,4,1))
```

returns the matrix:

2.00	2.00	-1.00
4.00	4.00	-2.00
-2.00	4.00	1.00

Note that the first matrix is not represented using the col function. This is because the col function returns a range, not a block, and our discussion above shows that the matrix would be interpreted as having one row rather than one column which is what is intended.

put into

The put into function places calculation results in a designated column on the worksheet. It operates faster than the equivalent equality relationship.

Syntax

```
put results into col(column)
```

The *results* argument can be either the result of an equation, function or variable. The *column* argument is either the column number of the destination column, or the column title, enclosed in quotes.

Data put into columns inserts or overwrites according to the current insert mode.

Example

To place the results of the equation y = data(1,100) in column 1, you can type `col(1) = y`. However, entering `put y into col(1)` runs faster.

random

This function generates a specified number of uniformly distributed numbers within the range. Rand and rnd are synonyms for the random function.

Syntax

```
random(number,seed,low,high)
```

The *number* argument specifies how many random numbers to generate.

The *seed* argument is the random number generation seed to be used by the function. If you want to generate a different random number sequence each time the function is used, enter 0/0 for the seed. If the seed argument is omitted, a randomly selected seed is used.

The *low* and *high* arguments specify the beginning and end of the random number distribution range. The low boundary is included in the range. If low and high are omitted, they default to 0 and 1, respectively.

Important: Function arguments are omitted from right to left. If you want to specify a high boundary, you must specify the low boundary argument first.

Example

The operation `random(50,0/0,1,7)` produces 50 uniformly distributed random numbers between 1 and 7. The sequence is different each time this random function is used.

real

The real function strips the real values from a complex block of numbers.

Syntax

```
real (range)
```

The *range* argument consists of complex numbers.

Example

If *x* = complex ({1,2,3,...,9,10}, {0,0,...,0}), the operation `real(x)` returns {1,2,3,4,5,6,7,8,9,10}, leaving the imaginary values out.

rgbcolor

The transform function rgbcolor takes arguments r, g, and b between 0 and 255 and returns the corresponding color to cells in the worksheet. This function can be used to apply custom colors to any element of a graph or plot that can use colors chosen from a worksheet column.

Syntax

```
rgbcolor(r,g,b)
```

The *r,g,b* arguments define the red, green, and blue intensity portions of the color. These values must be scalars between 0 and 255. Numbers for the arguments less than 0 or greater than 255 are truncated to these values.

Example

The operation `rgbcolor(255,0,0)` returns red.

The operation `rgbcolor(0,255,0)` returns green.

The operation `rgbcolor(0,0,255)` returns blue.

The following statements place the secondary colors yellow, magenta, and cyan into rows 1, 2, and 3 into column 1:

```
cell(1,1)=rgbcolor(255,255,0)
cell(1,2)=rgbcolor(255,0,255)
cell(1,3)=rgbcolor(0,255,255)
```

Shades of gray are generated using equal arguments. To place black, gray, and white in the first three rows of column 1:

```
cell(1,1)=rgbcolor(0,0,0)
 cell(1,3)=rgbcolor(255,255,255)cell(1,2)=rgbcolor(127,127,127)
```

root

Use the root function to find the roots of a function of one variable over a finite interval. In other words, the root function solves equations of the form f(x) = 0, where x is restricted to lie in a finite interval. This function also has the capability of finding certain values of the independent variable where the function is undefined, known as isolated singularities.

Syntax

```
rv = root(expr, variable, a, b, maxroots, type)
```

The *expr* argument defines the equations to solve. The expression can specify a range or list of functions so that more than one equation can be solved at a time. The equations are defined by setting each function in the expression list equal to zero. The *variable* argument is the symbol for the variable you are solving for in each specified equation. The same variable is used for all equations. The *a* and *b* arguments are the left and right endpoints, respectively, of the interval over which the root search takes place.

The *maxroots* and *type* arguments are optional. *Maxroots* is the maximum number of roots to compute for each specified function in the first argument. The default value is 1. *Type* is a number that specifies one of two types of output. If *type* = 0, then only roots will be returned. If *type* = 1, then only singularities will be returned. The default value is 0.

The Return Value, or *rv*, is the list or range of all of the roots that were found. The number of values returned will always be equal to *maxroots* for each function specified in the first argument. If fewer roots than *maxroots* are found, then the remaining values returned will be missing values. The reason for inserting the missing values is so the output of the different functions can be distinguished.

Helpful Tips

- Increasing the value of *maxroots* increases the chances of finding all of the solutions of the equation in the prescribed interval. It also increases the time required to complete the processing of the implicit function.
- When searching for multiple solutions to an equation, the *implicit* function partitions the interval that you specify into *maxroots* equally-spaced subintervals. It then searches each subinterval for exactly one solution. As a consequence, the implicit function may return fewer than *maxroots* solutions, even though the equation actually has *maxroots* or more solutions in the supplied interval. Ideally, to find all of the solutions to the equation over the interval from *a* to *b*, set maxroots to a value greater than (b -a)/delta, where delta estimates the closest distance between any solutions.
- The output of the implicit function is always sorted to give the roots in ascending order for each function in *expr*.

Example 1

This example uses a range of values to create a list of slightly modified equations. Two roots are found for each of the equations and the values are returned to the worksheet. Since *v* is a formal argument to a user-defined function, its value need not be initialized.

```
a=1
b=0
c=1
x= data(.1,.9,.1)
k(v)=a*x^2+b*x*v+c*v^2-1
col(2)=root(k(v),v,-10,10,2)
```

Example 2

Finds the two roots of the equation x^2+3*x-7=0. Note that *x* is initially set to 1 since each variable that is used in the transform language must be initialized unless it is a formal argument in a user-defined function as in the example above. The value that *x* is initially set to doesn't matter.

```
x=1
f=x^2+3*x-7
col(1)=root(f,x,-10,10,2)
```

Example 3

This third example is the same as above, but more direct.

```
x=1
col(2)=root(x^2+3*x-7,x,-10,10,2)
```

Example 4

This example uses range notation to enter multiple functions in the first argument of the root function. In this case, two roots are computed for each of three functions and the six values are returned to the worksheet.

```
x=1
f=x^2+3*x-7
g= cos((x+1)/5)
h=x*arctan(x)+.5*ln(x^2+1)-2
col(1)=root({f,g,h},x,-10,10,2)
```

It is assumed that the angular unit for this transform has been set to radians so that the value of *x* is interpreted in units of radians when finding the roots of *g*. In the output, the roots of function *f* are listed first, followed by the roots of the other two functions according to the order in which they appear in the list.

round

The round function rounds a number or range of numbers to the specified decimal places of accuracy. Values are rounded up or down to the nearest integer; values of exactly 0.5 are rounded up by default.

Syntax

```
round(numbers, places, mode)
```

The *numbers* argument can be a scalar or range of numbers. Any missing value or text string contained within a range is ignored and returned as the string or missing value.

If the *places* argument is negative, rounding occurs to the left of the decimal point. To round to the nearest whole number, use a places argument of 0.

The optional *mode* argument determines how rounding behaves if the last digit in the number is 5. If the value of mode is omitted or equal to zero, any number ending in 5 is rounded up. If the mode is not zero, then the number is rounded to the nearest even digit. In other words, if the digit before the 5 is odd, we round away from zero. If the digit before the 5 is even, we round towards 0.

Example 1

The operation `round(92.1541,2)` returns a value of 92.15. The operation `round(0.19112,1)` returns a value of 0.2. The operation `round(92.1541,-2)` returns a value of 100.0.

Example 2

The operation `round(1.25,1)` returns a value of 1.3. The operation `round(-1.45,1)` returns a value of -1.4. The operation `round(1.25,1,1)` returns a value of 1.2.

runavg

The runavg function produces a range of running averages, using a window of a specified size as the size of the range to be averaged. The resulting range is the same length as the argument range.

Syntax

```
runavg(range,window)
```

The *range* argument must be a single range (indicated with the {} brackets) or a worksheet column. Any missing value or text string contained within a range is replaced with 0.

If the *window* argument is even, the next highest odd number is used. The tails of the running average are computed by appending $\frac{(window-1)}{2}$ additional initial and final values to their respective ends of range.

Example

The operation `runavg({1,2,3,4,5},3)` returns {1.33,2,3,4,4.67}. The value of the window argument is 3, so the first result value is calculated as:

$$\frac{\frac{(3-1)}{2}+1+2}{3}$$

The second value is calculated as:

$$\frac{1+2+3}{3}$$

sin

This function returns ranges consisting of the sine of each value in the argument given.

This and other trigonometric functions can take values in radians, degrees, or grads. This is determined by the Trigonometric Units selected in the User-Defined Transform dialog box.

Syntax

```
sin(numbers)
```

The *numbers* argument can be a scalar or range.

If you regularly use values outside of the usual -2π to 2π (or equivalent) range, use the `mod` function to prevent loss of precision. Any missing value or text string contained within a range is ignored and returned as the string or missing value.

Example

If you choose **Degrees** as your Trigonometric Units in the Transform dialog box, the operation `sin({0,30,90,180,270})` returns values of {0,0.5,1,0,-1}.

sinh

This function returns the hyperbolic sine of the specified argument.

Syntax

```
sinh(numbers)
```

The *numbers* argument can be a scalar or range.

Like the circular trig functions, this function also accepts numbers in degrees, radians, or grads, depending on the units selected in the User-Defined Transform dialog box.

Example

The operation `x = sinh(col(3))` sets the variable x to be the hyperbolic sine of all data in column 3.

sinp

The sinp function automatically generates the initial parameter estimates for a sinusoidal functions using the FFT method. The three parameter estimates are returned as a vector.

Syntax

```
sinp(x range, y range)
```

The *x range* argument specifies the x variable, and the *y range* argument specifies the y variable. Any missing value or text string contained within one of the ranges is ignored and will not be treated as a data point. *x range* and *y range* must be the same size, and the number of valid data points must be greater than or equal to 3.

Tip: *sinp* is especially used to perform smoothing on waveform functions, used in determination of initial parameter estimates for nonlinear regression.

size

The size function returns a value equal to the total number of elements in the specified range, including all numbers, missing values, and text strings. Note that *size (X) 1/2 count (X) + missing (X)*.

Syntax

```
size(range)
```

The *range* argument must be a single range (indicated with the { } brackets) or a worksheet column.

sort

This function can be used to sort a range of numbers in ascending order, or a range of numbers in ascending order together with a block of data.

Syntax

```
sort(block,range)
```

The *range* argument can be either a specified range (indicated with the { } brackets) or a worksheet column. If the block argument is omitted, the data in range is sorted in ascending order.

Example 1

The operation `col(2) = sort(col(1))` returns the contents of column 1 arranged in ascending order and places it in column 2. To reverse the order of the sort, you can create a custom function:

```
reverse(x) = x[data(size(x),1)]
```

then apply it to the results of the sort. For example, `reverse(sort(x))` sorts range *x* in descending order.

Example 2

The operation:

```
block(3,1) = sort(block(1,1,2,size(col(2))),col(2))
```

sorts data in columns 1 and 2 using column 2 as the key column and places the sorted data in columns 3 and 4.

sqrt

The sqrt function returns a value or range of values consisting of the square root of each value in the specified range. Numerically, this is the same as {numbers}^0.5, but uses a faster algorithm.

Syntax

```
sqrt(numbers)
```

The *numbers* argument can be a scalar or range of numbers. Any missing value or text string contained within a range is ignored and returned as the string or missing value. For numbers < 0, sqrt generates a missing value.

Example

The operation `sqrt({-1,0,1,2})` returns the range {--,0,1,1.414}.

stddev

The stddev function returns the standard deviation of the specified range, as defined by:

$$s = \left[\frac{1}{n-1} \sum_{i=1}^{n} (x_i - x)^2 \right]^{\frac{1}{2}}$$

Syntax

```
stddev(range)
```

The *range* argument must be a single range (indicated with the {} brackets) or a worksheet column. Any missing value or text string contained within a range is ignored.

Example

For the range *x* = {1,2}, the operation `stddev(x)` returns a value of .70711.

stderr

The stderr function returns the standard error of the mean of the specified range, as defined by $\frac{s}{\sqrt{n}}$ where *s* is the standard deviation.

Syntax

```
stderr(range)
```

The *range* argument must be a single range (indicated with the { } brackets) or a worksheet column. Any missing value or text string contained within a range is ignored.

Example

For the range *x* = {1,2}, the operation `stderr(x)` returns a value of 0.5.

subblock

The subblock function returns a block of cells from within another previously defined block of cells from the worksheet. The subblock is defined using the upper left and lower right cells of the subblock, relative to the range defined by the source block.

Syntax

```
subblock (block, column 1, row 1, column 2, row 2)
```

The *block* argument can be a variable defined as a block, or a block function statement.

The *column 1* and *row 1* arguments are the relative coordinates for the upper left cell of the subblock with respect to the source block. The *column 2* and *row 2* arguments are the relative coordinates for the lower right cell of the subblock. All values within this range are returned. Operations performed on a block always return a block. If column 2 and row 2 are omitted, then the last row and/or column is assumed to be the last row and column of the source block.

All column and row arguments must be scalar (not ranges).

Example

For *x* = block (3,1,20,42) the operation `subblock (x,1,1,1,1)` returns cell (3,1) and the operation `subblock (x,5,5)` returns the block from cell (7, 5) to cell (20, 42).

sum

The function sum returns a range of numbers representing the accumulated sums along the list. The value of the number is added to the value of the preceding cumulative sum.

Because there is no preceding number for the first number in a range, the value of the first number in the result is always the same as the first number in the argument range.

Syntax

```
sum(range)
```

The *range* argument must be a single range (indicated with the { } brackets) or a worksheet column. Any text string or missing value contained within the range is returned as the string or missing value.

Example

For x = {2,6,7}, the operation `sum(x)` returns a value of {2,8,15}.

For y = {4,12,-6}, the operation `sum(y)` returns a value of {4,16,10}.

tan

This function returns ranges consisting of the tangent of each value in the argument given. This and other trigonometric functions can take values in radians, degrees, or grads. This is determined by the Trigonometric Units selected in the User-Defined Transform dialog box.

Syntax

```
tan(numbers)
```

The *numbers* argument can be a scalar or range.

If you regularly use values outside of the usual -2π to 2π (or equivalent) range, use the mod function to prevent loss of precision. Any missing value or text string contained within a range is ignored and returned as the string or missing value.

Example

If you choose Degrees as your Trigonometric Units in the transform dialog box, the operation `tan({0,45,135,180})` returns values of {0,1,-1,0}.

tanh

This function returns the hyperbolic tangent of the specified argument.

Syntax

```
tanh(numbers)
```

The *numbers* argument can be a scalar or range.

Example

The operation `x = tanh(col(3))` sets the variable *x* to be the hyperbolic tangent of all data in column 3.

tden

This function is the T-distribution's probability density function. It returns the value of the slope of the cumulative distribution function at the specified argument value.

Syntax

```
tden(x,n)
```

The *x* argument represents the independent variable and can either be a scalar or a range of numbers. If *x* is a range, then it must be defined by either using braces { } or by specifying a worksheet column. The *n* argument can be any positive integer and equals the degrees of freedom.

Example

The density function can be used to estimate the probability that the values of a T-distributed random variable T lie in a small interval. If T has 16 degrees of freedom, then to estimate the probability that the values of T lie between 1 and 1.1, multiply the density of T at 1 by the length of the interval .1:

```
tden(1,16) * .1 = .02346
```

tdist

This function is Student's T-distribution function. It returns the probability that a T-distributed random variable is less than a specified independent variable value.

A T-distributed random variable is defined as a scaled ratio of a standard normal variable and a chi-square variable. The *degrees of freedom* of a T-distributed variable is defined to be the degrees of freedom of the chi-square variable in the denominator.

This distribution is used in computing confidence intervals and for testing the homogeneity of populations for two groups of normally distributed observations.

Syntax

```
tdist(x,n)
```

The *x* argument represents the independent variable and can either be a scalar or a range of numbers. If *x* is a range, then it must be defined by either using braces { } or by specifying a worksheet column. The *n* argument can be any positive integer and equals the degrees of freedom.

Example

Suppose T is a T-distributed random variable with 14 degrees of freedom. To compute the probability that the absolute values of T exceed 2, we calculate:

```
P( |T| > 2 ) = P( T > 2) + P( T < -2)              = 2*P( T > 2)
 = 2*( 1 - P( T < 2) )              = 2*(1-tdist(2,14)) = .06529
```

This is a typical calculation that is used to test whether two normally distributed groups of observations have the same mean. In this context, the value 2 in our example is called the *critical value* and is equal to the absolute difference in the sample means of the two groups divided by the *pooled* standard deviation of the groups. The resulting probability, .06529, is called the *probability of significance*.

tinv

This function is the inverse of Student's T-distribution function. The probability that a T-distributed random variable is less than the return value is equal to the argument you specify.

Syntax

```
tinv(x,n)
```

The *x* argument can either be a scalar or a range of numbers. If *x* is a range, then it must be defined by either using braces { } or by specifying a worksheet column. Any scalar value for *x* represents a probability and so must be between 0 and 1. The *n* argument can be any positive integer and equals the degrees of freedom.

Example

Suppose a T-distributed random variable T has 23 degrees of freedom. .75 quartile of T, we calculate:

```
tinv(.75,23) = .68531
```

total

The function total returns a single value equal to the total sum of all numbers in a specified range. Numerically, this is the same as the last number returned by the sum function.

Syntax

```
total(range)
```

The *range* argument must be a single range (indicated with the { } brackets) or a worksheet column. Missing values and text strings contained within the range are ignored.

Example

For *x* = {9,16,7}, the operation `total(x)` returns a value of 32.

For *y* = {4,12,-6}, the operation `total(y)` returns a value of 10.

trp

The trp function generates the transpose of any matrix provided as a block.

Syntax

```
trp(block)
```

The *block* argument is a block of worksheet data containing numbers, unformatted ASCII text, blanks, missing values, or graphic cells. The function returns a block of data whose rows are the columns of the original block argument.

Example

For the matrix:

Group 1	**Group 2**	**Group 3**
2.00	4.00	2.00
1.00	3.00	2.00
3.00	0.00	1.00

in `block (1,1,3,4)` the operation

```
block (5,1) =trp (block (1,1,3,4))
```

generates the transpose matrix:

Table 3:

Group 1	2.00	1.00	3.00

Group 2	4.00	3.00	0.00
Group 3	2.00	2.00	1.00

in `block (5,1,8,3)`.

weibullden

This function is the Weibull distribution's probability density function. It returns the value of the slope of the cumulative distribution function at the specified argument value.

Syntax

```
weibullden(x,a,b)
```

The x argument represents the independent variable and can either be a scalar or a range of numbers. If x is a range, then it must be defined by either using braces { } or by specifying a worksheet column. Any value for x must be real. The a argument is any positive number and is the shape parameter. The b argument is any positive number and is the scale parameter.

Example

The density function can be used to estimate the probability that the values of a Weibull distributed random variable W lie in a small interval. If W has shape parameter equal to 1 and scale parameter equal to 1, then to estimate the probability that the values of W lie between 2 and 2.1, multiply the density of W at 2 by the length of the interval .1:

```
weibullden(2,1,1) * .1 = .013534
```

weibulldist

This function is the cumulative Weibull distribution function. It returns the probability that a Weibull distributed random variable is less than a specified independent variable value.

The Weibull distribution function describes the failure time distributions when the failure rate is assumed to increase as some power.

Syntax

```
weibulldist(x,a,b)
```

The *x* argument represents the independent variable and can either be a scalar or a range of numbers. If *x* is a range, then it must be defined by either using braces { } or by specifying a worksheet column. Any value for *x* must be real. The *a* argument is any positive number and is the shape parameter. The *b* argument is any positive number and is the scale parameter.

Example

Suppose a Weibull distributed random variable W has shape parameter equal to 1 and scale parameter equal to 1. To compute the probability that the values of W exceed 2, we calculate:

```
P( W > 2 ) = 1 – P( W < 2 ) = 1 – weibulldist(2,1,1) = .13534
```

weibullinv

This function is the inverse cumulative Weibull distribution function. The probability that a Weibull distributed random variable is less than the return value is equal to the argument you specify.

Syntax

```
weibullinv(x,a,b)
```

The *x* argument can either be a scalar or a range of numbers. If *x* is a range, then it must be defined by either using braces { } or by specifying a worksheet column. Any scalar value for *x* represents a probability and so must be between 0 and 1. The *a* argument is any positive number and is the shape parameter. The *b* argument is any positive number and is the scale parameter.

Example

Suppose a Weibull distributed random variable W has has shape parameter equal to 1 and scale parameter equal to 1. To calculate the tail of this distribution whose probability is .05, we need to find a number w such that P(W > w) = .05. This is the same as finding w such that P(W < w) = .95. Therefore, we calculate:

```
weibullinv(.95, 1, 1) = 2.99573
```

x25

The x25 function returns an interpolated value of the x data at

$$y_{\min} + \frac{range}{4}$$

in the ranges of coordinates provided, with optional Lowess smoothing. This is typically used to return the x value for the y value at 25% of the distance from the minimum to the maximum of smoothed data for sigmoidal shaped functions.

Syntax

```
x25(x range, y range, f )
```

The *x range* argument specifies the x variable, and the *y range* argument specifies the *y* variable. Any missing value or text string contained within one of the ranges is ignored and will not be treated as a data point. *x range* and *y range* must have the same size, and the number of valid data points must be greater than or equal to 3.

The optional *f* argument defines the amount of Lowess smoothing, and corresponds to the fraction of data points used for each regression. *f* must be greater than or equal to 0 and less than or equal to 1. ($0 \le f \le 1$). If *f* is omitted, no smoothing is used.

Example

For $x = \{0,1,2\}$, $y=\{0,1,4\}$, the operation

```
col(1)=x25(x,y)
```

places the *x* at

$$y_{\min} + \frac{range}{2}$$

as 1.00 into column 1.

x50

The x50 function returns an interpolated value of the x data at:

$$y_{\min} + \frac{range}{2}$$

in the ranges of coordinates provided, with optional Lowess smoothing. This is typically used to return the *x* value for the *y* value at 50% of the distance from the minimum to the maximum of smoothed data for sigmoidal shaped functions.

Syntax

```
x50(x range, y range, f )
```

The *x range* argument specifies the *x* variable, and the *y range* argument specifies the *y* variable. Any missing value or text string contained within one of the ranges is ignored and will not be treated as a data point. x range and y range must have the same size, and the number of valid data points must be greater than or equal to 3.

The optional *f* argument defines the amount of Lowess smoothing, and corresponds to the fraction of data points used for each regression. *f* must be greater than or equal to 0 and less than or equal to 1. ($0 \le f \le 1$). If *f* is omitted, no smoothing is used.

Example

For *x* = {0,1,2}, *y*={0,1,4}, the operation

```
col(1)=x50(x,y)
```

places the *x* at

$$y_{\min} + \frac{^{r}range}{2}$$

as 1.00 into column 1.

x75

The x75 function returns an interpolated value of the x data at:

$$y_{\min} + \frac{3 \cdot range}{4}$$

in the ranges of coordinates provided, with optional Lowess smoothing. This is typically used to return the x value for the y value at 75% of the distance from the minimum to the maximum of smoothed data for sigmoidal shaped functions.

Syntax

```
x75(x range, y range, f )
```

The *x range* argument specifies the *x* variable, and the *y range* argument specifies the *y* variable. Any missing value or text string contained within one of the ranges is ignored and will not be treated as a data point. x range and y range must have the same size, and the number of valid data points must be greater than or equal to 3.

The optional *f* argument defines the amount of Lowess smoothing, and corresponds to the fraction of data points used for each regression. *f* must be greater than or equal to 0 and less than or equal to 1. ($0 \le f \le 1$). If *f* is omitted, no smoothing is used.

Example

For *x* = {0,1,2}, y={0,1,4}, the operation

```
col(1)=x75(x,y)
```

places the *x* at

$$y_{\min} + \frac{3 \cdot range}{4}$$

as 2.00 into column 1.

xatymax

The xatymax function returns the interpolated x value at the maximum y value found, with optional Lowess smoothing.

Syntax

```
xatymax(x range, y range, f )
```

The *x range*argument specifies the *x* variable, and the *y* range argument specifies the *y* variable. Any missing value or text string contained within one of the ranges is ignored and will not be treated as a data point. *x range* and *y range* must have the same size, and the number of valid data points must be greater than or equal to 3. The optional *f* argument defines the amount of Lowess smoothing, and corresponds to the fraction of data points used for each regression. *f* must be greater than or equal to 0 and less than or equal to 1. $0 \le f \le 1$. If *f* is not defined, no smoothing is used.

Note: If duplicate *y* maximums are found *xatymax* will return the average value of all the *x* at *y* maximums.

Example

For x = {0,1,2}, y={0,1,4}, the operation

```
col(1)=xatymax(x,y)
```

places the x at the y maximum as 2.00 into column 1.

xwtr

The xwtr function returns value of x75-x25 in the ranges of coordinates provided, with optional Lowess smoothing.

Syntax

```
xwtr(x range, y range, f )
```

The *x range* argument specifies the x variable, and the *y range* argument specifies the y variable. Any missing value or text string contained within one of the ranges is ignored and will not be treated as a data point.*x range* and *y range* must have the same size, and the number of valid data points must be greater than or equal to 3.

The optional *f* argument defines the amount of Lowess smoothing, and corresponds to the fraction of data points used for each regression. *f* must be greater than or equal to 0 and less than or equal to 1. $0 \le f \le 1$. If *f* is omitted, no smoothing is used.

Example

For x = {0,1,2}, y={0,1,4}, the operation

```
col(1)=xwtr(x,y)
```

places the x75-x25 as double 1.00 into column 1.

Index

Special Characters

Numerics

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X