Arius®

Deterministic Formula Functions

Using Arius functions to build new arrays, exhibits, methods, and reports

Milliman

IT TAKES VISION

Milliman, Inc.
3424 Peachtree Road, NE Suite 1900
Atlanta GA 30326 USA

Tel   +1 800 404 2276
Fax   +1 404 237 6984

ActuarialSoftware.com

Table of Contents

1. Building Formulas

Arius has hundreds of system tables built in to store data, calculate diagnostics and actuarial methods, and store selections. In addition, you can create your own tables, referred to as “user defined” tables. User defined tables can be built from scratch or by copying another table and editing it as necessary. You can also copy an entire set of user defined tables from one project to another.

All user defined table activity occurs from the Object Library.

It might first help to understand table types:

- A **Data table** can be a triangle, row, column, or single cell (scalar), triangle index, or formula-driven assumption. It can be an entry field or a calculated field, and data tables can be referenced directly in formulas.
- An **Exhibit** is always a calculated triangle and is typically for development factors or other diagnostics. Exhibits are the only table type providing for a row of selected factors that can then be referenced in other formulas.
- **Methods and Reports** are columnar tables, where each column has a different calculation, and all columns are calculated fields. Methods are typically calculating an ultimate while Reports are typically summarizing information from Data, Exhibits, and Methods.

  The major difference between a Method and a Report is that Methods have one column designated as “ultimate” which can be referenced in other formulas. Columns in a report cannot be referenced directly from another table.

The easiest way to create a user defined table is to find an existing Arius table that is close to the type of table you need, copy it, and edit it as appropriate.

Copy or create a table

1. Copy an existing table
   - Highlight the table name and click **Copy**. Enter a new name and click **OK**.
   - From the User Defined folder highlight the table name and click **Edit**.
   - Update options as desired.

   Create a new table:

   - Click **New** then select the table type.
     - For Input and Exhibit tables, enter a name and set formats.
     - For Method and Report tables enter a name then click in a column to select column-specific formats and formulas.

2. Click Formula Editor

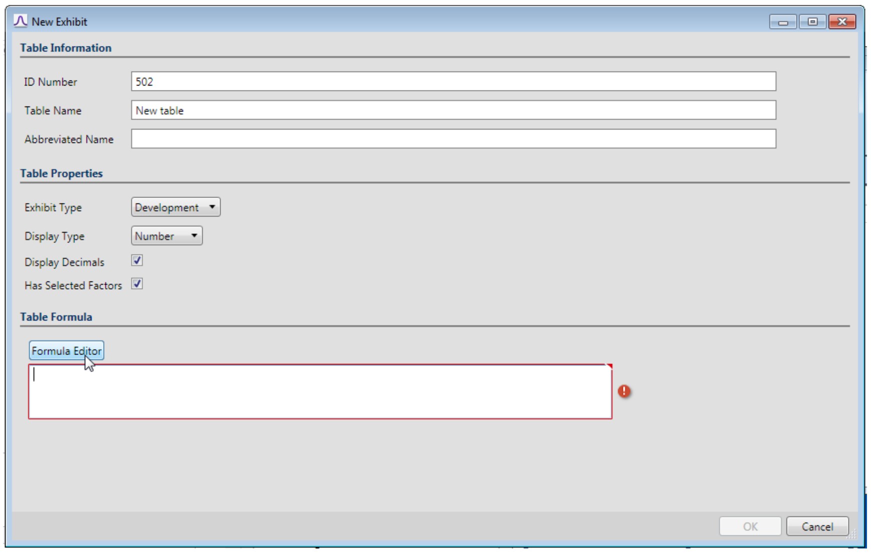


Table Information

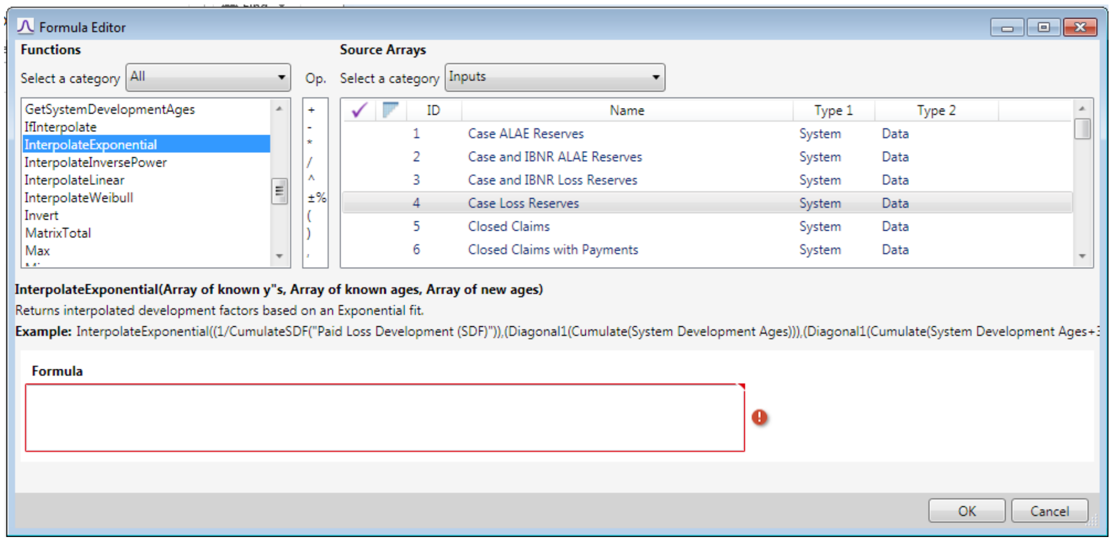
ID Number	502
Table Name	New table
Abbreviated Name	

Table Properties

Exhibit Type	Development
Display Type	Number
Display Decimals	✓
Has Selected Factors	✓

Table Formula

3. Click on a function. The screen displays the required syntax, the output and an example of each function.



ID	Name	Type 1	Type 2
1	Case ALAE Reserves	System	Data
2	Case and IBNR ALAE Reserves	System	Data
3	Case and IBNR Loss Reserves	System	Data
4	Case Loss Reserves	System	Data
5	Closed Claims	System	Data
6	Closed Claims with Payments	System	Data

InterpolateExponential(Array of known y"s, Array of known ages, Array of new ages)

Returns interpolated development factors based on an Exponential fit.

Example: InterpolateExponential((1/CumulateSDF("Paid Loss Development (SDF)")),(Diagonal1(Cumulate(System Development Ages))),(Diagonal1(Cumulate(System Development Ages+

2. Formula Functions

Table formulas are built by selecting the desired function, a mathematical operator and a table to complete the calculation. All formulas, operators and tables that you need to build a formula are provided in the Formula Editor.

This section describes the Formula Editor functions. Formula Editor Functions are selected to perform a particular calculation on designated data. The available functions are discussed below.

Note:

Formulas in Arius require 1) a space before and after parenthesis and 2) quotes around table names.

Abs

Returns the absolute value of an array of numbers

Input parameter:

- Any array, or Arius formulaic expression, for which you want the absolute values.

Output:

- An array containing the resulting absolute values (or values without their sign).
- Note: to take the absolute value of a cumulative array (A), use the formula: Decumulate (Abs (Cumulate (A)).

Example usage:

```
Sqrt ( Abs ( Decumulate ( Undevelop ( Cumulate ( "Paid Loss" ) ) ) ) )
```

BSAdjInc

Creates an exhibit or triangle of Berquist Sherman adjusted values (i.e. adjusted average Case Reserves).

Input parameters:

1. Unadjusted triangle array
2. Triangle index array

Output:

- An exhibit or triangle of Berquist Sherman adjusted values.

Example usage:

BSAdjInc ((Cumulate (“Case Loss Reserves”) / Cumulate (“Open Claims”)) * “Scaling Factor” , “Loss Trend Index – BS Adjusted Incurred”)

ConvertRowToCol

Converts a row object to a column object. Useful when working with asymmetrical files, such as when the length of development periods is “Quarter" and the length of exposure periods is "Year." Without this function, Arius will automatically treat row objects as a row of selected development factors and will compress the factors using multiplication. Use this function to override this behavior.

Input parameter:

- Any row of data

Output:

- Row displayed in a method or report in a column format

Example usage:

- Actual vs Expected reports:

  ConvertRowToCol (Invert (Shift (Invert (1/CumulateSDF ("Paid Loss Development (SDF)”)))))

ConvertSDFToCol

Converts a selected development factor row to a column object. Useful when working with asymmetrical files, such as when the length of development periods is "Quarter" and the length of exposure periods is "Year." This function allows you to perform other functions on the row object prior to the process of compressing the factors using multiplication to convert to column.

Input parameter:

- Any row of data or selected development factor array

Output:

- Row displayed in a method or report in a column format

Example usage:

- User defined reports:

  ConvertSDFToCol (SpreadTail (Shift (“Paid Loss Development (SDF)”)))

Cumulate

Sums each row of an array from left to right. This function is required in many formulas that use array data because Arius stores all balances in incremental form.

Input parameter:

- Any array or row of data

Output:

- Cumulated array, or a row (such as Payment Pattern) displayed in a method or report in column format

Example usage:

- Loss Development exhibits, Paid Loss Development exhibit:

```
Develop ( Cumulate ( “Paid Loss” ) )
```

CumulateSDF

Multiplies a row of data from right to left. The most common use of this function is to cumulate Selected Development Factors from age-to-age into age-to-ultimate factors (thus the function’s name).

Input parameter:

- Any row of data

Output:

- Cumulated row; will be displayed as a column in a method or report, with the most developed period first and the most recent period (for instance 12-24 period in annual data) displayed as the last item in the column.

Example usage:

- Loss methods, Paid Loss Development method, column #4:

```
CumulateSDF ( “Paid Loss Development (SDF)” )
```

Decumulate

Reverse of Cumulate (above); subtracts table data, working from right to left. Data in Arius is stored in incremental format. It may be necessary to perform certain calculations using cumulative data, and then decumulate the results for storage, especially in calculated arrays when tables are missing data (for instance when cells are blank). This function is primarily used when the resulting calculated data is being stored in the data file, as opposed to in a method or report wherein the data is always calculated at runtime.

Input parameter:

- Any array or row of data

Output:

- A decumulated array or row of data of similar shape and size.

Example usage:

- Case Reserves calculated from Paid and Incurred Loss:

```
Decumulate ( Cumulate ( “Incurred Loss” ) – Cumulate ( “Paid Loss” ) )
```

DecumulateCDF

Divides a row of data from left to right. The most common use of this function is to decumulate Cumulative Development Factors from age-to-ultimate into age-to-age factors (thus the function's name).

Input parameter:

- Any row or column of data

Output:

- A resulting row or column of data, displaying the decumulated values, will be displayed as a column in a method or report with the most developed period first and the most recent period (for instance, 12-24 period in annual data) displayed as the last item in the column.

Example usage:

- User defined reports (e.g., display implied patterns):

```
DecumulateCDF ( “Ultimate Loss” / Diagonal1 ( Cumulate ( “Paid Loss” ) ) )
```

Develop

Calculates age-to-age development factors from any array.

Input parameter:

- Any array

Output:

- Developed triangle

Example usage:

- Loss Development exhibits, Paid Loss Development exhibit:

  Develop (Cumulate (“Paid Loss”))

Diagonal

Retrieves the specified diagonal from any array. The system defaults to retrieve the latest (or most recent) diagonal, but you can specify the number parameter as well (e.g. 1 for most recent one, 2 for second one, etc.).

Input parameter:

- Any triangular array and specific diagonal

Output:

- A single column of data

Example usage:

- Loss methods, Paid Loss Development method, column #2:

  Diagonal1 (Cumulate (“Paid Loss”))

DiagonalAsRow

Retrieves the specified diagonal from any array and returns it as a row. The system defaults to retrieve the latest (or most recent) diagonal, but you can specify the number parameter as well (e.g. 1 for most recent one, 2 for second one, etc.).

Input parameter:

- Any triangular array and specific diagonal

Output:

- A single row of data

Example usage:

- User defined reports:

  DiagonalAsRow1 (Cumulate (“System Development Ages”))

FitExponential

Returns values along an exponential curve. Fits an exponential equation (using the method of least squares) to a set of data points or known_y's provided as the input parameter. The function will automatically exclude all blanks or invalid values (y <=0) in the fit.

Input parameter:

- Any column or row of data

Output:

- A resulting column or row of data, displaying the fitted values

Example usage:

- User-defined reports:

```
FitExponential ( "Average Loss" )
```

FitInversePower

Returns values along an inverse power curve. Fits an inverse power equation (using the method of least squares) to a set of data points or known_y's provided as the input parameter. The function will automatically exclude all blanks or invalid values (y <=0) in the fit.

Input parameter:

- Any column or row of data

Output:

- A resulting column or row of data, displaying the fitted values.

Example usage:

- User defined reports:

```
FitInversePower ( "Average Loss" )
```

FitLinear

Returns values along a straight line. Fits a linear equation (using the method of least squares) to a set of data points or known_y's provided as the input parameter. The function will automatically exclude all blanks.

Input parameter:

- Any column or row of data

Output:

- A resulting column or row of data, displaying the fitted values.

Example usage:

- User defined reports:

```
FitLinear ( "Average Loss" )
```

FitWeibull

Returns values along a Weibull curve. Fits a Weibull equation (using the method of least squares) to a set of data points or known_y's provided as the input parameter. The function will automatically exclude all blanks or invalid points (y<=0, y=1).

Input parameter:

- Any column or row of data

Output:

- A resulting column or row of data, displaying the fitted values.

Example usage:

- User defined reports:

```
FitWeibull ( "Average Loss" )
```

GetPriorSDF

Returns an array that represents the historical selected factors from the prior valuation date.

Input parameter:

- SDF row array

Output:

- An array representing the historical selected factors from the prior valuation date.

Example usage:

- ConvertRowToCol (GetPriorSDF (“Paid Loss Development (SDF)“))

GetProportionEarned

Returns the proportion of the exposure period earned based on the Exposure Period Type and given array of ages. The algorithm assumes uniform earning over the length of the exposure period. For example, in a 12 month exposure period, over 12 months for an Accident period and 24 months for a Policy period.

Input parameter:

- Any array

Output:

- Any array

Example usage:

- Proration of ultimate losses to reflect earned exposure for incomplete exposure periods:

```
Decumulate ( GetProportionEarned ( Cumulate ( ”System Development Ages” ) ) )
```

GetSystemDevelopmentAges

Returns the development ages (in months) corresponding to the file's data structure (e.g., shape and date parameters such as length of development period, number of development periods, ending month of first exposure period, first development age, and length of last calendar period).

Input parameter:

- None

Output:

- A triangular array, displaying the system ages (in months).

Example usage:

- System Development Ages array

    GetSystemDevelopmentAges ( )

IfInterpolate

Returns the first parameter of the function if TRUE and the second parameter if FALSE. IfInterpolate is TRUE if the last diagonal is a partial period as determined by the PROJECT SETTINGS | DATA STRUCTURE parameter for "Length of Last Calendar Period (in Months)."

Input parameters:

1. Any array
2. Any array

Output:

- Returns the first input parameter if TRUE, and returns the second input parameter if FALSE.

Example usage:

- Loss methods, Paid Loss Development, column #3:

    IfInterpolate ("Interpolated Paid Loss Development" , "Paid Loss Development (SDF)")

InterpolateExponential

Returns interpolated development factors based on the exponential curve fit and the three input parameters provided for known_y's, known_x's, and new_x's. In addition, calculations are based on the selection made under PROJECT SETTINGS | DATA STRUCTURE Exposure Period Type = {**Accident**, **Policy**, **Report** or **Underwriting**}.

Input parameters:

1. Array of known_y's, where known_y's represent your selected development pattern; e.g., this pattern can be described in terms of selected development factors, interpolated development factors, cumulative development factors, ratio to ultimates, etc.
2. Array of known_x's, where known_x's represent the development ages associated with the known_y's above; e.g., this array typically references a diagonal from the System Development Ages array.
3. Array of new_x's, where new_x's represent the development ages corresponding to the resulting interpolated y values; e.g., this array typically references a diagonal from the System Development Ages array.

Output:

- An array of new_y's, where new_y's represent the interpolated values returned from Arius' interpolation method.

Example usage:

- User defined reports; e.g., roll-forward to next quarter:

```
InterpolateExponential ( ( 1 / CumulateSDF ( “Interpolated Paid Loss Development” ) ) ,
( Diagonal1 ( Cumulate ( ”System Development Ages” ) ) ) , ( Diagonal1 ( Cumulate
( ”System Development Ages” ) ) ) +3 )
```

InterpolateInversePower

Returns interpolated development factors based on the inverse power curve fit and the three input parameters provided for known_y's, known_x's, and new_x's. In addition, calculations are based on the selection made under Project Settings | Data Structure Exposure Period Type = {Accident, Policy, Report or Underwriting}.

Input parameters:

1. Array of known_y's, where known_y's represent your selected development pattern; e.g., this pattern can be described in terms of selected development factors, interpolated development factors, cumulative development factors, ratio to ultimates, etc.
2. Array of known_x's, where known_x's represent the development ages associated with the known_y's above; e.g., this array typically references a diagonal from the "System Development Ages" array.
3. Array of new_x's, where new_x's represent the development ages corresponding to the resulting interpolated y values; e.g., this array typically references a diagonal from the "System Development Ages" array.

Output:

- An array of new_y's, where new_y's represent the interpolated values returned from Arius' interpolation method.

Example usage:

- User defined reports; e.g., roll-forward to next quarter:

```
InterpolateInversePower ( ( 1 / CumulateSDF ( "Interpolated Paid Loss Development" ) )  , (
Diagonal1 ( Cumulate ( "System Development Ages" ) ) ) , ( Diagonal1 ( Cumulate
( "System Development Ages" ) ) +3 ) )
```

InterpolateLinear

Returns interpolated development factors based on the linear curve fit and the three input parameters provided for known_y's, known_x's, and new_x's. In addition, calculations are based on the selection made under PROJECT SETTINGS | DATA STRUCTURE Exposure Period Type = {**Accident**, **Policy**, **Report** or **Underwriting**}.

Input parameters:

1. Array of known_y's, where known_y's represent your selected development pattern; e.g., this pattern can be described in terms of selected development factors, interpolated development factors, cumulative development factors, ratio to ultimates, etc.
2. Array of known_x's, where known_x's represent the development ages associated with the known_y's above; e.g. this array typically references a diagonal from the "System Development Ages" array.
3. Array of new_x's, where new_x's represent the development ages corresponding to the resulting interpolated y values; e.g., this array typically references a diagonal from the "System Development Ages" array.

Output:

- An array of new_y's, where new_y's represent the interpolated values returned from Arius' interpolation method.

Example usage:

- User defined reports; e.g., roll-forward to next quarter:

```
InterpolateLinear ( ( 1 / CumulateSDF ( "Interpolated Paid Loss Development" ) ) ,
( Diagonal1 ( Cumulate ( "System Development Ages" ) ) ) , ( Diagonal1 ( Cumulate
( "System Development Ages" ) ) +3 ) )
```

InterpolateWeibull

Returns interpolated development factors based on the Weibull curve fit and the three input parameters provided for known_y's, known_x's, and new_x's. In addition, calculations are based on the selection made under PROJECT SETTINGS | DATA STRUCTURE Exposure Period Type = {**Accident**, **Policy**, **Report** or **Underwriting**}.

Input parameters:

1. Array of known_y's, where known_y's represent your selected development pattern; e.g. this pattern can be described in terms of selected development factors, interpolated development factors, cumulative development factors, ratio to ultimates, etc.
2. Array of known_x's, where known_x's represent the development ages associated with the known_y's above; e.g. this array typically references a diagonal of the "System Development Ages" array.
3. Array of new_x's, where new_x's represent the development ages corresponding to the resulting interpolated y values; e.g. this array typically references a diagonal from the "System Development Ages" array.

Output:

- An array of new_y's, where new_y's represent the interpolated values returned from Arius' interpolation method.

Example usage:

- User defined reports; e.g. Roll-Forward to next quarter:

```
InterpolateWeibulll ( ( 1 / CumulateSDF ( “Interpolated Paid Loss Development” ) ) ,
( Diagonal1 ( Cumulate ( “System Development Ages” ) ) ) , ( Diagonal1 ( Cumulate
( ”System Development Ages” ) ) +3 ) )
```

Invert

Row data displayed in a column is normally displayed with the oldest data first and the most recent data last (for instance, a column of selected development factors in a development method). This function inverts the order, displaying least-developed amounts first.

Input parameter:

- A row of data

Output:

- A row of data, generally displayed as a column in a method or report

Example usage:

- Loss reports, Loss Development Patterns Based on Selected Development Factors report:

```
1 / ( Invert ( CumulateSDF ( “Paid Loss Development (SDF)” ) ) )
```

MatrixTotal

Returns only the final total column from a Payout Matrix (see Payout below).

Input parameter:

- Payout function (see Payout below)

Output:

- A single column of data

Example usage:

- Cash Flow reports, Indicated Loss Reserves Versus Present Value of Indicated Reserves report, column #2:

  MatrixTotal (Payout2 (Diagonal1 (Cumulate (“Indicated Case and IBNR Loss Reserves”)) , ”Loss Payment Pattern”) * (PVFactors (“Effective Interest Rate”)))

Max

Returns the maximum of two values.

Input parameters:

1. Any array of data
2. Any corresponding array of data

Output:

- Returns a corresponding array of data displaying the larger of the two values provided.

Example usage:

- User defined reports:

  Max (Column A, Column B)
  or
  Max (Max (Max (Column A, Column B) ,Column C) , Column D)

Min

Returns the minimum of two values.

Input parameters:

1. Any array of data
2. Any corresponding array of data

Output:

- Returns a corresponding array of data displaying the smaller of the two values provided.

Example usage:

- User defined reports:

```
Min ( Column A, Column B )
or
Min ( Min ( Min ( Column A, Column B ) ,Column C ) , Column D )
```

Payout

This function exists in Arius because it is required for ReservePro file imports. However, a new, more flexible version of this function, Payout2, has been implemented in Arius and it is suggested that the user utilize this new function in place of Payout.

Returns a matrix of future payments.

Input parameters:

1. A column of data to be paid out over time; often this is the Indicated Loss Reserve amount
2. The Loss Payment Pattern is also required, though it is not specifically a parameter passed to this function as part of the formula.

Output:

- A matrix, with a row for each exposure period and a column for each future period specified in the Loss Payment Pattern.

Example usage:

- Cash Flow reports, Future Payments of Indicated Loss Reserves report:

```
Payout ( “Ultimate Loss” – ( Diagonal1 ( Cumulate ( “Paid Loss” ) ) ) )
```

Payout2

Returns a matrix of future payments based on a given payment pattern. This function is similar to the Payout function defined above. However, this function provides flexibility in allowing the user to input a different payment pattern array than the default used in the Payout function defined above.

Input parameters:

1. A column of data to be paid out over time; often this is the Indicated Loss Reserve amount.
2. A resizable Payment Pattern row array.

Output:

- A matrix, with a row for each exposure period and a column for each future period specified in the Payment Pattern.

Example usage:

- Cash Flow reports, Future Payments of Indicated Loss Reserves report:

  Payout2 (Diagonal1 (Cumulate ("Indicated Case and IBNR Loss Reserves")) , "Loss Payment Pattern")

PVFactors

Returns a column of present value factors.

Input parameter:

- The Effective Interest Rate for each future period specified in any payment pattern array. The system assumes intra-year calculations for compound interest (e.g. monthly, quarterly, semiannual or annual as determined by the Length of Development Periods). For example, an annual interest rate of 5.0% would be entered as .0125 for a quarterly triangle.

Output:

- A column of PV factors with an entry for each future period specified in the Payment Pattern

Example usage:

- Cash Flow reports, Adjusted Payment Pattern report:

  PVFactors (“Effective Interest Rate”)

Round

Rounds all numbers in a resulting array to the specified number of decimal places.

Input parameters:

1. Any array, or another Arius formulaic expression
2. The number of decimal places to round to. Arius can accept values from 8 to -8.
   - If the second parameter is >0, round to that number of decimal places.
   - If the second parameter is zero, round to the nearest integer.
   - If the second parameter is <0, round to that number of places to the left of the decimal.

Note:

The formula wizard does not accept negative values. To round to the left of the decimal place, subtract the rounding parameter value from zero. As in the example shown, rather than entering -3, enter the expression 0-3.

Output:

- An array with resulting values rounded

Example usages:

- Round the cumulative Paid Loss triangle to 2 decimal places.

```
Round ( Cumulate ( "Paid Loss" ) , 2 )
```

- Round the Ultimate Loss array to the nearest thousand (e.g., an ultimate loss of 215,643.87 would round to 216,000).

```
Round ( "Ultimate Loss", 0-3 )
```

Shift

Moves all data one evaluation forward, for purposes of comparing prior data to the current evaluation.

Input parameter:

- Any array

Output:

- An array, with all amounts shifted by one evaluation

Example usage:

- Claims Ratios exhibits, Closed Claims to Prior Open Claims exhibit:

```
(Shift ( “Closed Claims” ) ) / ( Cumulate ( “Open Claims” ) )
```

SpreadTail

Spreads the tail factor out to the appropriate number of cells in the array when used in conjunction with the SHIFT function and a row of selected factors. This function assumes that the last cell of the array contains the tail factor and that the calculation uses the decay ratio associated with the previous two cells.

Input parameter:

- Any row of data or selected development factor array

Output:

- When used in conjunction with the SHIFT function, the output is the same row of selected development factors with the tail factor "spread"

Example usage:

- User defined reports

  ConvertRowToCol (SpreadTail (Shift (“Paid Loss Development (SDF)”)))

Sqrt

Returns the positive square root of an array of numbers

Input parameter:

- Any array, or Arius formulaic expression, for which you want the square roots. If a number is negative, a blank is returned so consider taking the absolute value of the array first, e.g., Sqrt (Abs (x))

Output:

- An array containing the resulting positive square roots.

  Note: to take the square root of a cumulative array (A), use the formula:
  Decumulate (Sqrt (Cumulate (A))

Example usage:

- User defined residual exhibit:

  Sqrt (Abs (Decumulate (Undevelop (Cumulate ("Paid Loss") , “Paid Loss Development (SDF)” , 1))))

Sum

Returns the sum of any row or column for use in user defined formulas.

Input parameter:

- Any row or column

Output:

- A row or column of data representing the "single-value" sum

Example usage:

- User defined table:

  "Ultimate Loss" / Sum ("Ultimate Loss")

Trend

Returns a column of trend indices given a single trend rate A. This function is commonly used when adjusting values to a common level and a constant trend rate can be assumed for all exposure years.

where j = total # of exposure periods (e.g., 10)
i = # of an exposure period in array (e.g., 1 - 10)

Input Parameter:

- Any column of data where the first cell represents a single trend rate percentage, e.g., 0.05.

Output:

- A column of data representing the trend indices, e.g., {1.276, 1.216, 1.158, 1.103, 1.050, 1.000}.

Example usage:

- User defined Generalized Cape Cod Methods; e.g., replicate Generalized Cape Cod Using Exposures and Paid Loss, but replace the formula in column 8 with the formula:

  Trend ("Trend Index – Cape Cod Using Exposures and Paid Loss")
  where the Trend Index array is a single value in the first cell representing a constant trend rate.

Undevelop

Returns a triangular array of cumulative “fitted” data resulting from developing backwards a given cumulative array, based on a selected “base” diagonal and selected development pattern

Input parameters:

1. Any triangular array, or Arius formulaic expression, representing the cumulative values you want to undevelop (or develop backwards) starting with a base diagonal.

   Note: the Cumulate function will need to be called before the array since Arius stores all balances in incremental form; e.g., Cumulate (“Paid Loss”) .

2. Any row of data or selected development factor array; e.g., “Paid Loss Development (SDF)”.
3. The integer >= 1 representing the diagonal to select as your base; e.g., where 1 is the latest diagonal, 2 is the previous diagonal, etc.

Output:

- An array containing the resulting cumulative “fitted” values.

  Note: the Decumulate function will need to be called on the resulting output since Arius stores all balances in incremental form.

Example usage:

- User defined:

```
Decumulate ( Undevelop ( Cumulate ( “Paid Loss” ) , “Paid Loss Development (SDF)”,
1 ) )
```

WeightedAvg

Returns the weighted average of the first parameter based on the weights provided by the second parameter.

Input parameters:

1. Any column of data corresponding to the value you want to weight.
2. Any column of data corresponding to the weights used to calculate the weighted average.
3. A column of data representing the “single-value” weighted average.

Example usage:

- User defined standard Cape Cod methods; e.g., replicate Generalized Cape Cod Using Exposures and Paid Loss, but remove column for Decay Ratio and replace formula for Expected Loss with:

```
WeightedAvg ( TrendDevLossRate, OnLevelExp*ExpPdWght )
```

WeightedAvgwithDecay

Returns the weighted average of the first parameter based on the weights provided by the second parameter and a “decay ratio”.

where j = total # exposure periods (e.g., 10)
i = # exposure period in array (e.g., 1 thru 10)

Input parameters:

1. Any column of data corresponding to the value you want to weight
2. Any column of data corresponding to the weights partially used to calculate the weighted average
3. Any column where the value in your first cell represents the decay ratio.

Output:

- A column of data representing the weighted average resulting from this “three-dimensional” weighting scheme

Example usage:

- Loss methods, Generalized Cape Cod Using Exposures and Paid Loss, Column #12

```
WeightedAvgWithDecay ( TrendDevLossRate, OnLevelExp*ExpPdWght, DecayRatio )
```

ZeroOutBlanks

Replaces any blank cells with zeroes during the calculation

Input parameter:

- Any array

Output:

- The same array, with zeroes in the place of blank cells

Example usage:

- Loss Ratio and Percent Change method, column #6:

```
UltPremiums * ( ZeroOutBlanks ( SelLossRatio ) + ZeroOutBlanks ( EstLossRatio ) )
```

3. Formula Editor Operators

Operators included in the Formula Editor perform specific mathematical operations. Each operator is described below.

+ – * /

Applies the appropriate mathematical function to the defined object. Objects on either side of the operator must have the same structure (for example, you cannot add a column-shaped table to a triangle).

Input parameter:

- Any input array or method/report column, or an amount specified in the formula

Output:

- An object of the same shape and structure as the two items in the formula

Example usage:

```
Diagonal1 ( Cumulate ( “Paid Loss” ) ) + Diagonal1 ( Cumulate ( “Paid ALAE” ) )
“Ultimate Loss” – Diagonal1 ( Cumulate ( “Paid Loss” ) )
Diagonal1 ( Cumulate ( “Paid Loss” ) * CumulateSDF ( ( "Paid Loss Development (SDF)” )
“Ultimate Loss” / ”Ultimate Premiums” * 100
```

^

Raises the selected object to a specified power

Input parameters:

1. Any input array or method/report column
2. A table of powers, or a power specified in the formula

Output:

- An object of the same shape and structure, raised to the specified power

Example usage:

- Column1 ^ 2 (squares the values in the first column)

±%

Applies percentage change factors to an amount to trend that amount into future periods. Percentage amounts must be entered as decimals (for example, 8% is entered as .08).

Input parameters:

1. Amount to be trended forward
2. A column of trend factors, entered as decimal values

Output:

- A column of trended amounts; calculated values begin on the row immediately following the row containing the balance to be trended, and continue through the last row of data

Example usage:

- Loss methods, Average Loss and Percentage Change, column #5:

  SelectedAverageLoss ±% AverageLossPercentageChange
  (This example takes the last amount entered in the Selected Average Loss column and trends it forward to all subsequent periods using the percent factors in the Percent Change column.)