

Math/Statistical Functions Available in *QuantVision*

The following table lists the functions available in the *QuantVision* system. In Aggregate and in Stock Specific with Aggregation functions, you may use the entire company database as your universe, or you may use a subset or a classification. If you use a classification name, the function name must be followed by a colon (e.g., **@DMN:Sector(PE,MCAP)**). If you use a company subset or a classification element, then enclose its name in curly braces (e.g., **@AVG{LGCAP}(PE)**).

For more details, see chapter 3 of the *QuantVision User's Guide*.

Function Call	Function Type	Function Description
@ABS(value)	Stock Specific	Returns the absolute value of the parameter.
@AVG(value)	Aggregate	Calculates the universe average of the parameter.
@AVGT[,](value)	Time Series	Calculates the average of the parameter over the specified time period.
@CEIL(value)	Stock Specific	Returns the ceiling value (i.e. the next highest whole number) of the parameter.
@CNT(value)	Aggregate	Calculates the total number of stocks in the universe with non-missing values for the parameter.
@CNTT[,](value)	Time Series	Calculates the total number of non-missing values of the parameter over the specified time period.
@DMN(variable, weight)	Stock Specific w/Aggregation	Calculates the universe demeaned value of the variable. Universe demeaning is accomplished by subtracting the universe average of the variable from the actual value. The weight parameter should contain a value to be used for weighting or a value of "1" for equal weighting. See chapter 8 of the <i>QuantVision User's Guide</i> for more information on demeaning.
@EXP(value)	Stock Specific	Calculates the exponential of the parameter (i.e., e raised to the value of the parameter value where e is equal to 2.71828).
@FLOOR(value)	Stock Specific	Returns the floor value (i.e. the next lowest whole number) of the parameter.

@FQEND()	Date	Returns the date of the next fiscal quarter end for a stock. This function depends on the fiscal year end attribute of a company that can be set in the <i>QV Universe</i> module. The returned value can be divided by a period constant (e.g. \$day, \$week) to yield a valid number.
@FYEND()	Date	Returns the date of the next fiscal year end for a stock. This function depends on the fiscal year end attribute of a company that can be set in the <i>QV Universe</i> module. @YEND() returns the date of the next calendar year end.
@IF(condition, TRUE, FALSE)	Stock Specific	This function allows you to specify a relational expression to be evaluated in the first parameter, followed by both a TRUE and FALSE evaluation paths. The evaluation paths should be variables or formulas to be evaluated in the event of a True/False condition to the first parameter. Some examples of the IF function are: @IF(PE<@AVG(PE), 1, 0) @IF(F1ESTS>3, PRICE/F1MN, \$na)
@LAG(value, date interval)	Stock Specific	Returns the value of parameter 1 as of the time period specified in parameter 2. This function is used when you need to compare data from one period to another. The second parameter should evaluate to a date interval (i.e. a period constant multiplied by a number). Some examples of the LAG function are: @LAG(PRICE, 28*\$day) @LAG(X*Y, 3*\$month)
@LOG(value)	Stock Specific	Returns the natural logarithm of the parameter.
@MAD(value)	Stock Specific w/Aggregation	Calculates the universe median absolute deviation (MAD) for the parameter.
@MADT[,](value)	Time Series	Calculates the MAD for the parameter over the specified time period.
@MAX(value)	Aggregate	Calculates the universe maximum value for the parameter.
@MAXT[,](value)	Time Series	Calculates the maximum value for the parameter over the specified time period.

@MED(value)	Aggregate	Calculates the universe median value for the parameter.
@MEDT[,](value)	Time Series	Calculates the median value for the parameter over the specified time period.
@MIN(value)	Aggregate	Calculates the universe minimum value for the parameter.
@MINT[,](value)	Time Series	Calculates the minimum value for the parameter over the specified time period.
@MOD(value, value)	Stock Specific	This function returns the remainder when the first parameter is divided by the second parameter.
@PMAX(value, value)	Stock Specific	Returns the maximum of the two parameters.
@PMIN(value, value)	Stock Specific	Returns the minimum of the two parameters.
@RANK(value, num fractiles)	Stock Specific w/Aggregation	This function allows you to rank the value in the first parameter by the number of fractiles specified in the second parameter. Ranking is performed across the universe. The expression @RANK(PRICE,5) would assign each stock a quintile rank (i.e., 1 through 5) based on it's price relative to all other stocks in the universe.
@RANKT[,](value, num fractiles)	Time Series	Calculates rank positions for the parameter over the specified time period. For example, the expression @RANKT[10,\$month](price,5) will look at the current period price value and rank it relative to the prior nine monthly price values by quintile (highest rank is 5, lowest rank is 1).
@RAW(variable)	Stock Specific	This function returns the unadjusted value of the parameter. If the value is not split sensitive or if no splits have occurred for a stock, the return value will be the same as if only the variable name was specified in the expression.
@ROUND(value)	Stock Specific	This function rounds the value to the nearest integer.
@SQRT(value)	Stock Specific	This function returns the square root of the value.
@STD(value)	Aggregate	Calculates the standard deviation of the value across the universe.
@STDT[,](value)	Time Series	Calculates the standard deviation of the value for a stock over the specified time period.

@STDZ(variable)	Stock Specific w/Aggregation	Standardizes the variable by subtracting the universe average from the raw value and then dividing the difference by the universe standard deviation.
@SUM(value)	Aggregate	Calculates the sum of the value across the universe.
@SUMT[,](value)	Time Series	Calculates the sum of the value for a stock over the specified time period.
@TODAY()	Date	Returns the date of the database period.
@WIN(variable)	Stock Specific w/Aggregation	Winsorizes the data to reduce the effect of outliers. This process iterates through the data and sets any data point outside the range: $\bar{x} - 3s < x < \bar{x} + 3s$ equal to the lower or upper bounds respectively.

Here are some practical examples of *QuantVision* computations:

@AVG(Bk_Prc)

Computes the average of variable Bk_Prc for the entire database universe.

@AVG{LGCAP}(Bk_Prc)

Computes the average of variable Bk_Prc for the LGCAP set of companies. LGCAP can be any subset created in **QV Port**, or a valid classification element (e.g., the Steel element of the Sector classification).

@AVG:Sector(Bk_Prc)

Computes the average of variable Bk_Prc for the sector that each company is in (e.g., the value shown for Microsoft would be the average Bk_Prc for its sector).

PE / @AVG{LGCAP}(PE)

Computes a relative PE figure. The numerator is the PE ratio for the company; the denominator is the average PE ratio for the companies in the LGCAP subset or classification element.

@DMN:Sector(Bk_Prc,MCAP)

Demean each company's Bk_Prc value relative to the cap-weighted Bk_Prc of its own sector. "Sector" must be a valid classification in your database. To equal weight, use 1 as the second argument in parentheses. See chapter 8 of the *QuantVision* User's Guide for more information on demeaning.

@DMN:Sector{LGCAP}(Bk_Prc,1)

Demean each company's Bk_Prc value in the subgroup LGCAP relative to its own sector's equal-weighted Bk_Prc. The subgroup LGCAP is created with **QV Port**, or is a valid classification element. See chapter 8 of the *QuantVision* User's Guide for more information on demeaning.

PE / @MEDT[26,\$week](PE)

Another relative PE calculation. The numerator is the company's PE ratio; the denominator is the 26-week median of the same company's PE ratio.

@WIN:Style(PE)

Winsorize the PE values by the Style classification (i.e., reduce the effects of data outliers). The Style classification might have Growth, Value elements.

@STDZ{Growth}(PE)

Standardize the PE values in the Growth element of some classification (e.g., Style), or the Growth subset.

@RANK(PE,10)

Rank PE values into deciles. Assign each stock in the selected universe a decile rank between 1 and 10, based on PE ratio. Companies in the top decile will have a value of 10. For quintiles, change the 10 to a 5. Any fractile will work.

@IF(@LAG(PE, 3 * \$month) < @MIN:Industry(PE), PE, \$na)

If the company's PE ratio three months ago is less than its industry's minimum PE ratio in the current period, then return that company's PE ratio. Otherwise, return "n/a".

@IF(@ACTIVE{LGCAP}(),PE,\$na)

If the company is a member of the company subset (or classification element) LGCAP, then return its value for PE; otherwise return a value of n/a.

PRICE - @LAG(PRICE, @TODAY() - @LAG(FQEND(), \$QUARTER))

Computes the quarter-to-date change in a variable called PRICE, in dollars and cents (not percentage).

@FLOOR((@TODAY() - @LAG(@YEND(), \$YEAR))/ \$DAY)

Computes the number of days we are into the current year.