SAS Performance Analytics Library

July 16, 2015

**Type** Package

**Title** Econometric tools for performance and risk analysis

**Version** 1.01

**Description**

A collection of econometric functions for performance and risk analysis. This library aims to aid practitioners and researchers in utilizing research in analysis of return streams. In general, it is most tested on returns (rather than price) data on a regular scale, but most functions will work with irregular return data as well. This package aims to replicate the Performance Analytics package available in R with a few minor tweaks to improve functionality, but otherwise adhering very closely to R Performance Analytics.

**Copyright** © 2015

**Authors** Dominic Pazzula

Carter Johnston

**Date/Publication** 2015-07-16

# SAS topics documented:

[ActivePremium……………………………………………………………………………………………………………………………7](#name_activePremium)

[Adjusted\_SharpeRatio…………………………………………………………………………………………………………………8](#name_Adjusted_SharpeRatio)

[Appraisal\_Ratio…………………………………………………………………………………………………………………………..9](#name_appraisalRatio)

[Average\_Drawdown……………………………………………………………………………………………………….…………11](#name_averageDrawdown)

[Average\_Length…….……………………………………………………………………………………………………….…………12](#name_AverageLength)

[Average\_Recovery……………………………………………………………………………………………………….…………13](#name_AverageRecovery)

[BetaC](#name_BetaCo)[oMoments](#name_BetaCo)…………………………………………………………………………………………………………….…………14

[Bull\_B](#name_BullBearbeta)[ear\_beta..…………………………………………………………………………………………………………….…………15](#name_BullBearbeta)

[Burke\_Ratio…...…………………………………………………………………………………………………………….…………16](#name_BurkeRatio)

[Calmar\_Ratio…...…………………………………………………………………………………………………………….…………18](#name_CalmarRatio)

[CAPM\_alpha\_beta…………………………………………………………………………………………………………….………19](#name_CAPMalphabeta)

[CAPM\_epsilon………………………………………………………………………………………………………………….……….20](#name_CAPMepsilon)

[CAPM\_JensenAlpha………………………………………………………………………………………………………………….22](#name_CAPMJensenalpha)

[Centered\_moments………………………………………………………………………………………………………………….23](#name_CenteredMoments)

[Chart\_Autoregression…………………………………………………………………………………………………………….24](#ChartAutoregression_TOC)

[Chart\_Correlation……….…………………………………………………………………………………………………………….26](#ChartCorrelation_TOC)

[Chart\_CumulativeReturns……………………………………………………………………………………………………….27](#ChartCumulativeReturns_TOC)

[Chart\_Drawdown.…………………………………………………………………………………………………………………….28](#name_ChartDrawdown)

[Chart\_ECDF…..……………………………………………………………………………………………………………………….30](#name_ChartECDF)

[Chart\_Histogram……………………………………………………………………………………………………………………….30](#name_ChartHistogram)

[C](#name_ChartRegression)[hart\_Regression…………………………………………………………………………………………………………………….32](#name_ChartRegression)

[Chart\_RelativePerformance…………………………………………………………………………………………………….33](#name_ChartRelativePerformance)

[Chart\_Scatter……………………………………………………………………………………………………………………………34](#name_ChartScatter)

[CoMoments………………………………………………………………………………………………………………………………36](#name_CoMoments)

[Crea](#name_CreateHash)[te\_Hash……………………………………………………………………………………………………………………………37](#name_CreateHash)

[Download\_ff3……………………………………………………………………………………………………………………………38](#name_Downloadff3)

[Drawdowns….……………………………………………………………………………………………………………………………38](#name_Drawdowns)

[Drawdown\_Deviation………………………………………………………………………………………………………………39](#name_DrawdownDeviation)

[Fama\_Beta………………………………………………………………………………………………………………………………..41](#name_famabeta)

[Find\_Drawdowns.……………………………………………………………………………………………………………………..42](#name_FindDrawdowns)

[Geo\_Mean………………………………………………………………………………………………………………………………..43](#name_GeoMean)

[Get](#name_GetStocks)[\_Stocks………………………………………………………………………………………………………………………………..44](#name_GetStocks)

[Information\_Ratio…………………………………………………………………………………………………………………….45](#name_InformationRatio)

[Martin\_Ratio…….](#name_MartinRatio)[………………………………………………………………………………………………………………………47](#name_MartinRatio)

[Max\_drawdown………………………………………………………………………………………………………………………48](#name_MaxDrawdown)

[Mean\_Abs\_Deviation………………………………………………………………………………………………………………49](#name_MeanAbsDeviation)

[MSquared…………………………………………………………………………………………………………………………………50](#name_MSquared)

[Pain\_Index………………………………………………………………………………………………………………………………..51](#name_PainIndex)

[Pain\_Ratio………………………………………………………………………………………………………………………………..52](#name_PainRatio)

[Prices………………………………………………………………………………………………………………………………………..54](#name_prices)

[Return\_Annualized……………………………………………………………………………………………………………………54](#name_Return_Annualized)

[Return\_Calculate………………………………………………………………………………………………………………………56](#name_Return_Calculate)

[Return\_Centered……………………………………………………………………………………………………………………...57](#name_Return_Centered)

[Return\_Cumulative…………………………………………………………………………………………………………………..58](#name_Return_Cumulative)

[Return\_Excess……………………………………………………………………………………………………………………….….59](#name_Return_Excess)

[Sharpe\_Ratio………………………………………………………………………………………………………………………..…..60](#name_SharpeRAtio)

[SharpeRatio\_Annualized………………………………………………………………………………………………………..…61](#name_SharpeRatioAnnualized)

[Simple\_Normalize……………………………………………………………………………………………………………..…63](#name_SimpleNormalize)

[Simple\_Normalize\_By………………………………………………………………………………………………………..…63](#name_SimpleNormalizeBy)

[Sort\_Drawdowns.](#name_SortDrawdowns)[……………………………………………………………………………………………………………………..64](#name_SortDrawdowns)

[Specific\_Risk……………………………………………………………………………………………………………………………..65](#name_SpecificRisk)

[Standard\_Deviation…………………………………………………………………………………………………………………..66](#name_StandardDeviation)

[StdDev\_Annualized………………………………………………………………………………………………………………..…68](#name_StdDevAnnualized)

[Ster](#name_SterlingRatio)[ling\_Ratio……….………………………………………………………………………………………………………………..…69](#name_SterlingRatio)

[Systematic\_Risk………………………………………………………………………………………………………………..………70](#name_SystematicRisk)

[Table\_Annualized\_Returns…………………………………………………………………………………………..………….72](#name_tableAnnualizedReturns)

[Table\_AutoCorrelations…………………………………………………………………………..……………………….………73](#name_tableAutoCorrelations)

[Table\_CalendarReturns…………………………………………………………………………………………………….………74](#name_tableCalendarReturns)

[Table\_](#name_tableCAPM)[CAPM……..………………………………………………………………………………………………………………..……75](#name_tableCAPM)

[Table\_Correlation………………………………………………………………………………………………………………..……76](#name_tableCorrelations)

[Table\_Distributions……………………………………………………………………………………………………………..……77](#name_tableDistributions)

[Table\_Drawdowns……..](#name_TableDrawdowns)[……………………………………………………………………………………………………….…….78](#name_TableDrawdowns)

[Table\_DrawdownsRatio…………………………………………………………………………………………………….…….79](#name_tableDrawdownsRatio)

[Table\_HigherMoments……………………………………………………………………………………………………….…….80](#name_tableHigherMoments)

[Table\_InformationRatio……………………………………………………………………………………………………………81](#name_tableInformationRatio)

[Table\_SpecificRisk…………………………………………………………………………………………………………………….83](#name_tableSpecificRisk)

[Table\_Stats…………………………………………………………………………………………………………………………….…84](#name_tableStats)

[Table\_Variability…………………………………………………………………………………………………………………….…85](#name_tableVariability)

[TrackingError…………………………………………………………………………………………………………………….………86](#name_trackingError)

[Treynor\_Ratio………………………………………………………………………………………………………………….………87](#name_TreynorRatio)

[Ulcer\_Index…………………………………………………………………………………………………………………….………89](#name_UlcerIndex)

**SAS Performance Analytics Library**

Econometric tools for performance and risk analysis.

**Description**

Working for the Financial Risk Group, the creators of the SAS Performance Analytics macro library saw an opportunity: to recreate the performance analytics package found in R in SAS to implement performance analysis tools needed for the FRG platform. Performance Analytics provides a SAS package of econometric functions for performance and risk analysis of financial instruments or portfolios. This packages aims to aid practitioners and researchers in using the latest research for analysis of a returns series.

We created this library to include functionality that appears in the R Performance Analytics package, which is taken from academic literature on performance analysis and risk. These tools had no functional equivalent in SAS previously. Generally, this package requires return data rather than price data. However, price data can be quickly converted to returns data with the use of the macro [Return\_Calculate](#name_Return_Calculate). Almost all of the macros in this library will work with annual, quarterly, monthly, or daily frequency. In the following summary, we attempt to provide an overview of the capabilities provided by SAS Performance Analytics. We hope that the accompanying library and documentation can fill the void when it comes to financial risk and performance analytics tools available to an analyst using the SAS system.

With the growing accessibility to alternative assets to the individual investor, demand has become ever higher for research and analysis tools in performance analytics. The simple tools that were appropriate in a relative investment world now seem inappropriate for investment returns in the current context. Risk measurement, which is inseparable from performance assessment, has become multi-dimensional and multi-moment all the while attempting to answer a very simple question: “What is my risk?” Portfolio construction and risk budgeting are then two sides of the same coin: optimizing a portfolio by maximizing return while minimizing volatility. With the increasing availability of complicated alternative investment strategies to investors, and the state of near perfect information, an engaging debate about performance analysis and evaluation is as crucial as ever.

Performance analytics does not guarantee a perfect portfolio immune to all risk. However, what it does offer is an accretion of evidence, organized to assist a decision maker in answering a specific question on a particular asset or portfolio. Using such tools to uncover information and ask better questions will create a better informed investor. Performance measurement starts with returns. However, the normalization inherent in calculating returns can be deceiving. It is important that returns be standardized because this “price per unit of investment” standardization is useful in comparing opportunity costs and because of the standardization’s useful statistical qualities. As a result, the Performance Analytics library focuses on standardized returns rather than prices [See [Return\_calculate](#name_Return_Calculate) for converting net asset values or prices into returns, either discrete or log based]. Many papers and theories refer to “excess returns”, or risk premium: we implement a simple function for aligning time series and calculating these excess returns in [Return\_excess](#name_Return_Excess). Returns and risk may be annualized as a way to simplify comparison over longer or unequal time periods. Although it requires a bit of estimating, such aggregation is popular because it offers a reference point for easy comparison. Examples of this estimation can be found in [Return\_annualized](#name_Return_Annualized), [StdDev\_annualized](#name_StdDevAnnualized), and [SharpeRatio\_annualized](#name_SharpeRatioAnnualized). Basic measures of performance tend to treat returns as independent observations. In this case, the entirety of the SAS base is applicable to such analysis. Some basic statistics are collected in [table.Stats](#name_tableStats).

These types of summary statistics and tables provide the bulk of the information an investor may want to analyze, and provides an organized way to view results of potentially thousands of periodic return data. Usually these statistics are the most “readable” when organized into a table of related statistics assembled for a particular purpose. A common offering of past returns organized by month and cumulated by calendar year is usually presented as a table, such as in [table\_CalendarReturns](#name_tableCalendarReturns). Adding benchmarks or peers alongside the annualized data is helpful for comparing returns in calendar years. Examples of other tables for comparison of related groupings of statistics discussed in this documentation:

[table.Stats](#name_tableStats) provides Basic statistics and stylized facts

[table.AnnualizedReturns](#name_tableAnnualizedReturns) Annualized return, standard deviation, and Sharpe ratio [table.CalendarReturns](#name_tableCalendarReturns) Monthly and calendar year return table

[table.Correlation](#name_tableCorrelations) Comparison of correlations and significance statistics

[table.Autocorrelation](#name_tableAutoCorrelations) The first six autocorrelation coefficients and significance [table.HigherMoments](#name_tableHigherMoments) Higher co-moments and beta co-moments

[table\_Distributions](#name_tableDistributions) provides distribution statistics

[table\_InformationRatio](#name_tableInformationRatio) Provides the information ratio as well as the tracking error and annualized tracking error.

[table\_SpecificRisk](#name_tableSpecificRisk) Table of specific risk, systematic risk, and total risk.

[table\_Variability](#name_tableVariability) Table of variability statistics from a returns data set.

Modern Portfolio Theory (MPT), although somewhat ironically outdated, is the collection of tools and techniques by which a risk-averse investor may construct an “optimal” portfolio. It was pioneered by Harry Markowitz in 1952 and encompasses CAPM, the efficient market hypothesis, and all forms of quantitative portfolio construction and optimization. The Capital Asset Pricing Model (CAPM), initially developed by William Sharpe in 1964, provides a justification for passive or index investing by proposing that assets that are not on the efficient frontier will either rise or fall in price until they are. The [CAPM­ alpha](#name_CAPMalphabeta) is the degree to which the asset’s returns are not due to the return that could be captured from the market as a whole. Conversely, the [CAPM beta](#name_CAPMalphabeta) describes the portions of the returns of the asset that could be directly attributed to the returns of a passive investment in the benchmark asset. CAPM is a market equilibrium model or a general equilibrium theory of the relation of prices to risk, but it is usually applied to partial equilibrium portfolios, which can create (sometimes serious) problems in valuation. The performance premium provided by an investment over a passive strategy (the benchmark) is provided by the [active premium](#name_activePremium), which is the investment’s annualized return minus the benchmark’s annualized return. A closely related measure is the [Tracking Error](#name_trackingError), which measures the unexplained portion of the investment’s performance relative to a benchmark. The [Information Ratio](#name_InformationRatio) of an investment in a MPT or CAPM framework is the Active Premium divided by the Tracking Error. The Information Ratio may be used to rank investments in a relative fashion. Research shows that relative rankings across multiple pricing methodologies may be positively correlated with each other and with expected returns. This is quite an important finding because it shows that multiple methods of predicting returns and risk which have underlying measures and factors which are not directly correlated to another measure or factor will still producesimilar quantile rankings. While analyzing an asset or portfolio using the performance analytics tools in this library does not guarantee greater returns, verifying the asset or portfolio over multiple measures for evidence will help prove a positive investment decision for the rational investor.

While we acknowledge that the library is currently incomplete, we hope to continue to append the library as much as we can and in due time find its functionality competitive with that of R Performance Analytics.

**Authors**

Dominic Pazzula

Carter Johnston

**[ActivePremium](#ActivePremium_TOC)** Active Premium or Active Return

**Description**

The return on an investment’s annualized return minus the benchmark’s annualized return.

**Usage**

**%**ActivePremium(Returns, BM=)

%ActivePremium(Returns, BM=, scale=, method=)

%ActivePremium(Returns, BM=, scale=, method=, dateColumn=, outData=)

**Arguments**

Returns - required. Names the data set containing time series data of asset returns.

BM- required. Specifies the benchmark asset or index in the returns data set.

Scale- The number of periods per year used in the annualized calculation.

Method- option to annualize using DISCRETE or LOG chaining. {DISCRETE, LOG} [Default= DISCRETE]

dateColumn- specifies the date column in the data set. [Default= Date]

outData- specifies the name of the output active premium. [Default= active\_premium]

**Author**

Dominic Pazzula, Carter Johnston

**See Also**

[CoMoments](#name_CoMoments)

**Examples**

%let dir=C:\SVN\SAS\_Perf\_Anly;

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***ActivePremium***(prices, BM= SPY);

/\*Or\*/

/\*%ActivePremium(prices, BM= DOW, scale= 252, method= DISCRETE, dateColumn= Date, outData= active\_premium);\*/

**[Adjusted\_SharpeRatio](#adjustedSharpe_TOC)** Adjusted Sharpe Ratio of the return distribution

**Description**

Adjusted Sharpe ratio was introduced to adjust for skewness and kurtosis by incorporating a penalty for negative skewness and excess kurtosis.

**Usage**

**%**Adjusted\_SharpeRatio(Returns)

%Adjusted\_SharpeRatio(Returns, Rf =, scale=)

%Adjusted\_SharpeRatio(Returns, Rf=, scale=, VARDEF=, dateColumn=, outData=)

**Arguments**

Returns - required. Names the data set containing time series data of asset returns.

Rf- the value or variable representing the risk free rate of return.

Scale- The number of periods per year used in the annualized calculation. [Default= 1]

VARDEF- Optional. Specify the variance divisor, DF, degree of freedom, n-1; N, number of observations, n. {N, DF} Default= DF.

dateColumn- specifies the date column in the data set. [Default= Date]

outData- specifies the name of the output Adjusted Sharpe ratios [Default= adjusted\_SharpeRatio]

**Author**

Dominic Pazzula, Carter Johnston

**See Also**

[Sharpe\_Ratio](#name_SharpeRAtio)  [SharpeRatio\_annualized](#name_SharpeRatioAnnualized)

**Examples**

%let dir=C:\SVN\SAS\_Perf\_Anly;

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***Adjusted\_SharpeRatio***(prices);

/\*Or\*/

/\*%Adjusted\_SharpeRatio(prices, Rf= 0.01/252, scale= 252, dateColumn= Date, outData= adjusted\_SharpeRatio);\*/

**[Appraisal\_Ratio](#appraisalRatio_TOC)** Appraisal ratio of the return distribution

**Description**

Appraisal ratio is the Jensen’s alpha adjusted for specific risk. The numerator is divided by specific risk instead of total risk.

**Details**

Modified Jensen’s alpha is Jensen’s alpha divided by beta.

Alternative Jensen’s alpha is Jensen’s alpha divided by systematic risk.

**Usage**

**%**Appraisal\_Ratio(Returns, BM=, option=)

%Appraisal\_Ratio(Returns, BM=, Rf=, scale=, option=)

%Appraisal\_Ratio(Returns, BM=, Rf=, scale=, option=, method=, VARDEF=, dateColumn=, outData=)

**Arguments**

Returns - required. Names the data set containing time series data of asset returns.

BM- required. Specifies the benchmark asset or index in the returns data set.

Rf- the value or variable representing the risk free rate of return.

Scale- The number of periods per year used in the annualized calculation. [Default= 1]

Option- required. {APPRAISAL, MODIFIED, ALTERNATIVE}. Choose “appraisal” to calculate the appraisal ratio, “modified” to calculate the modified Jensen’s alpha, or “alternative” to calculate alternative Jensen’s alpha.

Method- option to annualize Jensen’s alpha using DISCRETE chaining or LOG chaining. {DISCRETE, LOG} [Default= DISCRETE]

VARDEF- Optional. Specify the variance divisor, DF, degree of freedom, n-1; N, number of observations, n. {N, DF} Default= DF.

dateColumn- specifies the date column in the data set. [Default= Date]

outData- specifies the name of the output Appraisal ratios [Default= appraisal\_ratio]

**Author**

Dominic Pazzula, Carter Johnston

**See Also**

[CAPM\_JensenAlpha](#name_CAPMJensenalpha)

**Examples**

%let dir=C:\SVN\SAS\_Perf\_Anly;

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***Appraisal\_Ratio***(prices, BM= SPY, Rf= 0, option= appraisal);

/\*Or\*/

/\*%Appraisal\_Ratio(prices, BM= SPY, Rf= 0.01/252, scale= 252, option= modified, method= DISCRETE, dateColumn= Date, outData= appraisal\_ratio);\*/

**[Average\_Drawdown](#averageDrawdown_TOC)** calculate average drawdown

**Description**

Calculate average depth of observed drawdowns over entire period.

**Details**

The average depth is calculated as:

Where d is total number of drawdowns and Dt is the tth drawdown.

**Usage**

%Average\_Drawdown(returns)

%Average\_Drawdown(returns, method=, dateColumn=, outData)

**Arguments**

returns - Required. Data Set containing returns.

method - Optional. Specifies either DISCRETE or LOG chaining method {DISCRETE, LOG}. [Default=DISCRETE]

dateColumn - Optional. Date column in Data Set. [Default=DATE]

outData - Optional. Output Data Set with average drawdowns. [Default="Avg\_DD"]

**Author**

Qiyuan Yang

**Examples**

%let dir=C:\SVN\SAS\_Perf\_Anly;

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***Average\_Drawdown***(prices);

/\*Or\*/

/\*%Average\_Drawdown(prices, method=LOG, dateColumn=DATE, outData=MyAveDrawdown);\*/

**[Average\_Length](#AverageLength_TOC)** calculate average length of drawdown

**Description**

Finds the arithmetic mean value of drawdown length. Multiple assets can be calculated at the same time.

**Details**

In order to calculated average lengths from multiple asset drawdowns, Find\_Drawdowns() is not called in this macro. Instead, the drawdown length of each asset is recalculated. In R function AverageLength, the compounding method for return is defaulted as geometric and it is not an input option to change. In this macro, user can choose the compounding method.

**Usage**

%Average\_Length(returns)

%Average\_Length(returns, method=, dateColumn=, outData)

**Arguments**

returns - Required. Data Set containing returns.

method - Optional. Specifies either DISCRETE or LOG chaining method {DISCRETE, LOG}. [Default=DISCRETE]

dateColumn - Optional. Date column in Data Set. [Default=DATE]

outData - Optional. Output Data Set with average length. [Default="AverageLength"]

**Author**

Ruicheng Ma

**Examples**

%let dir=C:\SVN\SAS\_Perf\_Anly;

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***Average\_Length***(prices);

/\*Or\*/

/\*%Average\_Length(prices, method=LOG, dateColumn=DATE, outData=MyAverageLength);\*/

**[Average\_Recovery](#AverageRecovery_TOC)** calculate average recovery of drawdown

**Description**

Finds the arithmetic mean value of drawdown recovery. Multiple assets can be calculated at the same time.

**Usage**

%Average\_Recovery(returns)

%Average\_Recovery(returns, method=, dateColumn=, outData=)

**Arguments**

returns - Required. Data Set containing returns.

method - Optional. Specifies either DISCRETE or LOG chaining method {DISCRETE, LOG}. [Default=DISCRETE]

dateColumn - Optional. Date column in Data Set. [Default=DATE]

outData - Optional. Output Data Set with average recovery. [Default="AverageRecovery"]

**Author**

Ruicheng Ma

**Examples**

%let dir=C:\SVN\SAS\_Perf\_Anly;

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***Average\_Recovery***(prices);

/\*Or\*/

/\*%Average\_Recovery(prices, method=LOG, dateColumn=DATE, outData=MyAverageRecovery);\*/

[**BetaCoMoments**](#betaCo_TOC) Functions to calculate systematic or beta co-moments of return series

**Description**

Calculate higher co-moment betas, or ‘systematic’ variance, skewness, and kurtosis matrices.

Beta Covariance is equivalent to Covariance over Variance, Beta Coskewness is equivalent to Coskewness over skewness, and Beta CoKurtosis is equivalent to CoKurtosis over Kurtosis.

**Usage**

**%**BetaCoMoments(Returns)

%BetaCoMoments(Returns, dateColumn=)

%BetaCoMoments(Returns, dateColumn=, outBetaCoVar=, outBetaCoSkew=, outBetaCoKurt=)

**Arguments**

Returns - required. Names the data set containing time series data of asset returns.

dateColumn- specifies the date column in the data set. [Default= Date]

outBetaCoVar- specifies the name of the output Beta Covariance matrix. [Default= BetaM2]

outBetaCoSkew- specifies the name of the output Beta Coskewness matrix. [Default= BetaM3]

outBetaCoKurt- specifies the name of the output Beta Cokurtosis matrix. [Default= BetaM4]

**Author**

Dominic Pazzula, Carter Johnston

**See Also**

[CoMoments](#name_CoMoments)

**Examples**

%let dir=C:\SVN\SAS\_Perf\_Anly;

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***BetaCoMoments***(prices);

/\*Or\*/

/\*%BetaCoMoments(prices, dateColumn= Date,outBetaCoVar=BetaM2, outBetaCoSkew= BetaM3, outBetaCoKurt=BetaM4);\*/

**[Bull\_Bear\_beta](#BullBearbeta_TOC)** calculate bull/bear beta from CAPM model

**Description**

The beta from CAPM model is now divided into two classes: bull beta when market return is positive, and bear beta when market return is negative. Bull/bear beta measures portfolio sensitivity to the market under different market performances.

**Usage**

%Bull\_Bear\_beta(returns=, BM=);

%Bull\_Bear\_beta(returns=, BM=, Rf=, dateColumn=, outData=);

**Arguments**

returns - Required. Data Set containing returns with option to include risk free rate variable.

BM - Required. Specifies the variable name of benchmark asset or index in the returns data set.

Rf - Optional. The value or variable representing the risk free rate of return. [Default=0]

dateColumn - Optional. Date column in Data Set. [Default=DATE]

outData - Optional. Output Data Set of asset Alphas and Betas. [Default= "alphas\_and\_betas"]

**Author**

Qiyuan Yang

**See Also**

CAPM\_alpha\_beta.

**Example**

%let dir=C:\SVN\SAS\_Perf\_Anly;

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***Bull\_Bear\_beta***(prices, BM=IBM);

/\*Or\*/

/\*%Bull\_Bear\_beta(prices, BM=IBM, Rf=0, dateColumn=DATE, outData=MyBullBear);\*/

**[Burke\_Ratio](#BurkeRatio_TOC)** calculate Burke Ratio

**Description**

Calculate Burke Ratio with the option to output modified Burke Ratio.

**Details**

Burke Ratio is similar to Calmar Ratio and Sterling Ratio where risk is expressed in form of drawdowns. Burke Ratio has the difference of asset return and risk free rate as its numerator and square root of the sum of squared drawdowns as its denominator. It can thus be calculated as:

In above formulas, d is total number of drawdowns, n is total number of observations from the data set, and Dt is the tth drawdown.

**Usage**

%Burke\_Ratio(returns);

%Burke\_Ratio(returns, Rf=, scale=, asset=, method=, modified=, dateColumn=, outData=);

**Arguments**

returns - Required. Data Set containing return.

Rf - Optional. the value or variable representing the risk free rate of return. [Default=0]

scale - Optional. Number of periods in a year {any positive integer, ie daily scale= 252, monthly scale= 12, quarterly scale= 4}. [Default=1]

method - Optional. Specifies either DISCRETE or LOG chaining method {DISCRETE, LOG}. [Default=DISCRETE]

modified - Optional. Option to calculate modified Burke Ratio. {TRUE, FALSE} [Default=FALSE]

dateColumn - Optional. Date column in Data Set. [Default=DATE]

outData - Optional. Output Data Set with Burke Ratios. [Default="BurkeRatio"]

**Author**

Qiyuan Yang

**See Also**

[Calmar\_Ratio](#name_CalmarRatio),[Sterling\_Ratio](#name_SterlingRatio), [Sharpe\_Ratio](#name_SharpeRAtio)

**Example**

%let dir=C:\SVN\SAS\_Perf\_Anly;

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***Burke\_Ratio***(prices);

/\*Or\*/

/\*%Burke\_Ratio(prices, Rf=0, scale=252, method=LOG, modified=TRUE, dateColumn=DATE, outData=MyBurkeRatio);\*/

**[Calmar\_Ratio](#CalmarRatio_TOC)** calculate Calmar Ratio

**Description**

Calmar Ratio measures an asset’s return against its drawdown risk. Similar to Sharpe Ratio and Sterling Ratio, it is one of the statistics that measures return vs. risk.

**Details**

Calmar Ratio is calculated as annualized return divided by maximum drawdown.

**Usage**

%Calmar\_Ratio(returns);

%Calmar\_Ratio(returns, scale=, method=, dateColumn=, outData=);

**Arguments**

returns - Required. Data Set containing returns.

scale - Optional. Number of periods in a year {any positive integer, ie daily scale= 252, monthly scale= 12, quarterly scale= 4}. [Default=1]

method - Optional. Specifies either DISCRETE or LOG chaining method {DISCRETE, LOG}. [Default=DISCRETE]

dateColumn - Optional. Date column in Data Set. [Default=DATE]

outData - Optional. Output Data Set with Calmar ratios. [Default="CalmarRatio"]

**Author**

Qiyuan Yang

**See Also**

[Drawdowns](#name_Drawdowns), [max\_drawdown](#name_MaxDrawdown), [Sharpe\_Ratio](#name_SharpeRAtio), [Sterling\_Ratio](#name_SterlingRatio)

**Example**

%let dir=C:\SVN\SAS\_Perf\_Anly;

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***Calmar\_Ratio***(prices);

/\*Or\*/

/\*%Calmar\_Ratio(prices, scale=4, method=LOG, dateColumn=DATE, outData=MyCalmarRatio);\*/

**[CAPM\_Alpha\_Beta](#CAPMalphabeta_TOC)** calculate single factor model (CAPM) alpha and beta

**Description**

This macro calculates values of Alpha and Beta as defined by CAPM (single factor model), effectively combining the wrapper functions of CAPM.alpha and CAPM.beta in the R performance analytics package.

**Details**

“Alpha” purports to be the measure of a manager’s skill by measuring the portion of the manager’s returns not attributable to “Beta”, or the portion of performance attributable to a benchmark asset or index.

It should be noted that the classical CAPM model has been almost completely discredited by academics, however, it is a good example of a simple single factor model comparing an asset to an arbitrary benchmark.

The CAPM Beta is the beta of an asset to the variance and covariance of an initial portfolio. It is used to determine diversification potential. Beta is often thought of as the slope of the regression line used to determine the risk premium of a returns time series. Alpha is thought of as the intercept of this regression line.

Beta can be calculated as:

**Usage**

**%**CAPM\_Alpha\_Beta (Returns, BM=, Rf=);

%CAPM\_Alpha\_Beta (Returns, BM=, Rf=, dateColumn=, outData=);

**Arguments**

Returns - required. Names the data set containing time series data of asset returns.

BM- required. Names variable in the data set containing returns of the benchmark asset or index.

Rf- value or variable representing the risk free rate of return.

dateColumn- specifies the date column in the data set. [Default= Date]

outData- output data set containing values of alpha and beta. [Default= alphas\_and\_betas]

**Author**

Dominic Pazzula, Carter Johnston

**See Also**

[CAPM\_Epsilon](#name_CAPMepsilon)

**Examples**

%let dir=C:\SVN\SAS\_Perf\_Anly;

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***CAPM\_Alpha\_Beta***(prices, BM= SPY, Rf= **0.01**/**252**);

/\*Or\*/

/\*%CAPM\_Alpha\_Beta(prices, BM= SPY, Rf= IBM, dateColumn= Date, outData= alphas\_and\_betas);\*/

**[CAPM\_Epsilon](#CAPMepsilon_TOC)** Regression epsilon of the return distribution

**Description**

The regression epsilon is an error term measuring the vertical distance between the return predicted by the equation and the real result as defined by the capital asset pricing model.

**Details**

The regression epsilon is given by the following formula:

Where is the regression alpha, is the regression beta, is the portfolio return and b is the benchmark return.

**Usage**

%CAPM\_Epsilon(Returns, BM=, Rf=, scale=);

%CAPM\_Epsilon(Returns, BM=, Rf=, scale=, dateColumn=, outData=);

**Arguments**

Returns - required. Names the data set containing time series data of asset returns.

BM- required. Names variable in the data set containing returns of the benchmark asset or index.

Rf- value or variable representing the risk free rate of return.

Scale- required. Specifies the number of periods in one year. [Daily= 252, Monthly= 12, Quarterly= 4]

dateColumn- specifies the date column in the data set. [Default= Date]

outData- output data set containing values of epsilon. [Default= epsilon]

**Author**

Dominic Pazzula, Carter Johnston

**See Also**

[CAPM\_Alpha\_Beta](#name_CAPMalphabeta)

**Examples**

%let dir=C:\SVN\SAS\_Perf\_Anly;

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***CAPM\_Epsilon***(prices, BM= SPY, Rf= **0.01**/**252**);

/\*Or\*/

/\*%CAPM\_Epsilon(prices, BM= SPY, Rf= IBM, scale= 252, dateColumn= Date, outData= epsilon);\*/

[CAPM\_JensenAlpha](#CAPMJensenalpha_TOC) Jensen’s alpha of the return distribution

**Description**

The Jensen’s alpha is the intercept of the regression equation in the Capital Asset Pricing Model and is in effect the excess return adjusted for systematic risk.

**Details**

Jensen alpha is calculated as:

.

where is the risk free rate, is the regression beta, is the portfolio return and b is the benchmark return.

**Usage**

%CAPM\_JensenAlpha(Returns, BM=);

%CAPM\_JensenAlpha(Returns, BM=, Rf=, scale=, method=, dateColumn=, outData=);

**Arguments**

Returns - required. Names the data set containing time series data of asset returns.

BM- required. Specifies the benchmark asset or index.

Rf- the value or variable specified as the risk free rate of return.

Scale- The number of periods per year used in the annualized calculation. [Default= 1]

Method- option to implement DISCRETE or LOG chaining when annualizing.

dateColumn- specifies the date column in the data set. [Default= Date]

outData- output data set containing values of Jensen alphas. [Default= Jensen\_Alpha]

**Author**

Dominic Pazzula, Carter Johnston

**See Also**

[CAPM\_alpha\_beta](#name_CAPMalphabeta)

**Examples**

%let dir=C:\SVN\SAS\_Perf\_Anly;

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***CAPM\_JensenAlpha***(prices, BM= SPY);

/\*Or\*/

/\*%CAPM\_JensenAlpha(prices, BM= SPY, Rf= 0.01/252, scale= 252, method= DISCRETE, dateColumn= Date, outData= Jensen\_Alpha);\*/

**[Centered\_Moments](#CenteredMoments_TOC)** calculate centered moments

**Description**

This macro is used internally by SAS Performance Analytics to calculate centered moments for a multivariate distribution as well as the standardized moments of a portfolio distribution. However, it can be called independently for those who wish to calculate centered moments directly.

**Details**

The *n*-th centered moment is calculated as:

.

Centered\_Moments returns values of the centered variance, centered skewness, and centered kurtosis in separate tables.

**Usage**

%Centered\_Moments(Returns);

%Centered\_Moments(Returns, dateColumn=, outCenteredVar=, outCenteredSkew=, outCenteredKurt=);

**Arguments**

Returns - required. Names the data set containing time series data of asset returns.

dateColumn- specifies the date column in the data set. [Default= Date]

outCenteredVar- output data set containing values of centered variance. [Default= centered\_Var]

outCenteredSkew- output data set containing values of centered skewness. [Default= centered\_Skew]

outCenteredKurt- output data set containing values of centered kurtosis. [Default= centered\_Kurt]

**Author**

Dominic Pazzula, Carter Johnston

**See Also**

[Return\_Centered](#name_Return_Centered)

**Examples**

%let dir=C:\SVN\SAS\_Perf\_Anly;

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***Centered\_Moments***(prices);

/\*Or\*/

/\*%Centered\_Moments (prices, dateColumn= Date, outCenteredVar= centered\_Var, outCenteredSkew= centered\_Skew, outCenteredKurt= centered\_Kurt);\*/

[**Chart\_Autor****egression**](#name_ChartAutoregression)  A series of auto-regression charts

**Description**

Create a series of auto-regression charts for analysis using a return data set. This macro is in tandem with chart.ACFplus from R library. Besides the default setting of plotting ACF and PACF plots, this macro includes many other plots for users.

**Usage**

%Chart\_Autoregression(returns, asset=, lag=)

%Chart\_Autoregression(returns, asset=, lag=, title= , ALL=, dateColumn=)

**Arguments**

returns - Required. Data Set containing returns.

asset - Required. Specifies the variable or asset to be plotted.

lag - Required. Specifies the amount of lags to plot in each chart.

title - Optional. Title for Charts. Default= AutoRegression Analysis for &asset

ALL - Optional. Option to plot all charts available via Proc Timeseries for analysis. Default= FALSE

ACF - Optional. Option to plot an ACF chart for the specified lag. Default= TRUE

PACF - Optional. Option to plot a PACF chart for the specified lag. Default= TRUE

WN - Optional. Option to plot White Noise charts for the specified lag. Default= FALSE

IACF - Optional. Option to plot Inverse ACF charts for the specified lag. Default= FALSE

RESIDUAL - Optional. Option to plot Residual charts for the specified lag. Default= FALSE

SeasonalAdjusted - Optional. Option to plot a Seasonal adjusted chart for the specified lag. Default= FALSE

SeasonalComponent - Optional. Option to plot a Seasonal component chart for the specified lag. Default= FALSE

SeasonalCycle - Optional. Option to plot a Seasonal cycle chart for the specified lag. Default= FALSE

TrendComponent - Optional. Option to plot a trend component chart for the specified lag. Default= FALSE

TrendCycleComponent - Optional. Option to plot a trend cycle component chart for the specified lag. Default= FALSE

TrendCycleSeasonal - Optional. Option to plot a Seasonally adjusted trend cycle chart for the specified lag. Default= FALSE

dateColumn - Optional. specifies the date column for returns in the data set. Default= Date

**Author**

Dominic Pazzula, Carter Johnston

**Examples**

%let dir=C:\SVN\SAS\_Perf\_Anly;

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***Chart\_Autoregression***(prices,asset=IBM,lag=5,ALL=TRUE);

/\*Or\*/

/\*%Chart\_Autoregression(prices, asset=IBM, lag=5, title=My Autogression, ACF=FALSE, PACF=FALSE, WN=TRUE, RESIDUAL=TRUE, dateColumn=DATE);\*/

[**Chart\_Co****rrelation**](#name_ChartCorrelation) Correlation matrix of returns

**Description**

Create a table showing the correlation of returns between every two assets.

**Usage**

%Chart\_Correlation(returns, title=, histogram=)

%Chart\_Correlation(returns, title=, histogram=, histogramDensity=, color=, symbol=, size= )

**Arguments**

returns - Required. Data Set containing returns.

title - Required. Title for histogram. Default= Portfolio Asset Correlations

histogram - Optional. Option to insert histograms for each asset along the diagonal of the plot matrix. Default= FALSE

histogramDensity - Optional. Selects a type of density to overlay on histograms along the diagonal. Default= normal {Normal, Kernel}

color - Optional. Change the color of the scatter plot points. Default= cornflowerblue

symbol - Optional. Change the symbol of the scatter plot points. Default= circle

See list of possible symbols at SAS product documentation (markerattrs symbol)

size - Optional. Change the size (in pixels) of the plot points. Default= 6

ellipse - Option to add a predictive ellipse to scatter plots. Default= FALSE {True, False}

ellipseType - Optional. If ellipse is overlayed, specifies type. Default= predicted {mean, predicted}

alpha - Optional. If ellipse is overlayed, specifies value of alpha for predictive bands. Default= 0.05

dateColumn - Optional. Specifies the date column for returns in the data set. Default= Date

**Author**

Dominic Pazzula, Carter Johnston

**Examples**

%let dir=C:\SVN\SAS\_Perf\_Anly;

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***Chart\_Correlation***(prices);

/\*Or\*/

/\*%Chart\_Correlation(prices, title= Correlation Chart, histogram= TRUE, histogramDensity=kernel, color=red, symbol= star, size=8, ellipse=TRUE, ellipseType=mean, dateColumn=DATE);\*/

**[Chart\_CumulativeReturns](#name_ChartCumulativeReturns)** plot cumulative return

**Description**

Create a chart of cumulative returns for all asset in the given data set.

**Usage**

%Chart\_CumulativeReturns(returns)

%Chart\_CumulativeReturns(returns, title=, method=, WealthIndex=, grid=, Interval=1, dateColumn=)

**Arguments**

returns - Required. Data Set containing returns.

title - Optional. Title for chart. [Default= Cumulative Returns]

method - Optional. Specifies either DISCRETE or LOG chaining method {DISCRETE, LOG}.

[Default=DISCRETE]

WealthIndex - Optional. Specifies that the value of a dollar in the first sample is $1, therefore charting the value of the returns per dollar over time. [Default= FALSE] {TRUE, FALSE}

grid - Optional. Overlay grid lines on the returns axis. [Default= TRUE]

Interval - Optional. Specifies the frequency of grid lines overlayed on the returns axis. [Default= 1 (100%)]

dateColumn - Optional. Specifies the date column for returns in the data set. [Default= Date]

**Author**

Dominic Pazzula, Carter Johnston

**See Also**

[Return\_Cumulative](#name_Return_Cumulative)

**Examples**

%let dir=C:\SVN\SAS\_Perf\_Anly;

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***Chart\_CumulativeReturns***(prices);

/\*Or\*/

/\*%Chart\_CumulativeReturns(prices, title= My Cumulative Returns, method=LOG, WealthIndex=TRUE, grid=FALSE, dateColumn=DATE);\*/

**[Chart\_Drawdown](#ChartDrawdown_TOC)** chart of drawdowns

**Description**

Create a chart that displays multiple asset drawdowns through time.

**Usage**

%Chart\_Drawdown(returns, asset=)

%Chart\_Drawdown(returns, asset=, method=, title=, grid=, Interval=, linecolor=, legend\_pos=, dateColumn=)

**Arguments**

returns - Required. Data Set containing returns.

asset - Required. Name of the variable to plot drawdown chart for. Asset names are separated by space. {e.g. asset=IBM GE DOW}

method - Optional. Specifies either DISCRETE or LOG chaining method {DISCRETE, LOG}. [Default=DISCRETE]

title - Optional. Title for chart. [Default= Drawback Chart for &asset]

grid - Optional. Overlay grid lines on the returns axis. [Default= TRUE]

Interval - Optional. Specifies the frequency of grid lines overlayed on the returns axis. [Default= -0.1 (-10%)]

linecolor - Optional. Specifies the color of the lines. See SAS COLOR NAMES for reference. {e.g. for three assets, linecolor=GOLD BLACK RED} [Default: automatically assigned by SAS]

legend\_pos - Optional. Position of key legend. See SAS KEYLEGEND statement for more information. [Default= BOTTOMLEFT]

dateColumn - Optional. Specifies the date column for returns in the data set. [Default= DATE]

**Author**

Ruicheng Ma

**Examples**

%let dir=C:\SVN\SAS\_Perf\_Anly;

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***Chart\_Drawdown***(prices, asset=IBM GE DOW);

/\*Or\*/

/\*%Chart\_Drawdown(prices, asset=IBM GE DOW, method=LOG, title= my chart, grid=TRUE, Interval=-0.2, linecolor=GOLD OLIVE SALMON, legend\_pos=BOTTOM, dateColumn=DATE);\*/

**[Chart\_ECDF](#ChartECDF_TOC)** CDF of asset return in comparison with a Normal CDF

**Description**

Create a chart for the CDF of asset return in comparison with a fitted normal density CDF.

**Usage**

%Chart\_ECDF(returns)

%Chart\_ECDF(returns, title=, dateColumn=)

**Arguments**

returns - Required. Data Set containing returns.

title- Optional. Title for chart. [Default= Empirical CDF]

dateColumn - Optional. Specifies the date column for returns in the data set. [Default= Date]

**Author**

Dominic Pazzula, Carter Johnston

**Examples**

%let dir=C:\SVN\SAS\_Perf\_Anly;

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***Chart\_ECDF***(prices);

/\*Or\*/

/\*%Chart\_ECDF(prices, title= CDF Comparison, dateColumn=DATE);\*/

**[Chart\_Histogram](#ChartHistogram_TOC)** Histogram of returns

**Description**

Create a histogram of returns with the option to overlay a density curve to show approximate fit.

**Usage**

%Chart\_Histogram(Returns, asset, title=);

%Chart\_Histogram(Returns, asset, scale=, title=, bindwidth=, density=, color=, dateColumn=)

**Arguments**

Returns - required. Names the data set containing time series data of asset returns.

Asset- required. Specifies the variable for which to calculate the histogram of returns.

Scale- Optional. Specifies whether the y-axis should go by probability or frequency. {count, percent, proportion}. [Default= count]

Title- give title for histogram [Default= &asset]

bindwidth - Optional. Specifies the range of returns to select for each bar. [Default= 0.001]

Density- Option to overlay a normal density on the histogram for normal approximation. If true, [Density= Density]

color- Optional. Change the color of the histogram bins. [Default= cornflowerblue]

densitycolor- Optional. Change the color of the density line. [Default= red]

histogramTransparency- Optional. Change the transparency of the histogram bins. [Default= 0.8]

keepOutliers- Optional. Delete outlier returns from the histogram within the range of Q1- 1.5IQR and Q3+1.5IQR. [Default= TRUE]. {TRUE, FALSE}

qqplot - Optional. Display a QQ Plot in addition to the histogram. [Default= FALSE]. {TRUE,FALSE}

rug - Optional. Display a fringe plot overlayed onto the histogram. [Default= FALSE]. {TRUE,FALSE}

dateColumn- specifies the date column in the data set. [Default= Date]

**Author**

Dominic Pazzula, Carter Johnston

**Examples**

%let dir=C:\SVN\SAS\_Perf\_Anly;

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***Chart\_Histogram***(prices, asset= IBM);

/\*Or\*/

/\*%Chart\_Histogram(prices, asset= IBM, scale=percent, title= My Histogram, density= TRUE, dateColumn= Date);\*/

**[Chart\_Regression](#ChartRegression_TOC)** Chart of simple regression

**Description**

Create a chart of simple regression between an asset return and a benchmark, given a return data set. It has the option to overlay a Loess line, and the option to apply linear or quadratic regression.

**Usage**

%Chart\_Regression(returns, xvar=, yvar=)

%Chart\_Regression(returns, xvar=, yvar=, title=, ExcessReturns=, loess=, dateColumn=)

**Arguments**

returns - Required. Data Set containing returns with option to include risk free rate variable.

xvar - Required. Specifies the variable or asset to be plotted on the x-axis.

yvar - Required. Specifies the variable or asset to be plotted on the y-axis (a benchmark asset).

title - Optional. Title for Scatter Plot. [Default= xvar versus yvar Regression Plot]

ExcessReturns - Optional. Option to plot returns in excess of the benchmark or a risk free rate. {TRUE, FALSE} [Default= FALSE]

Rf - Optional. If excessReturns is true, then specifies the risk free rate as a number or as a benchmark asset. {Rf= 0.05, Rf= SPY} [Default= 0]

grid- Optional. Overlay a grid aligned with the points on the x and y axis. {TRUE,FALSE} [Default= TRUE]

transparency - Optional. Specifies the level of transparency for data symbols. [Default= 0.35]

color - Optional. Change the color of the scatter plot points. [Default= cornflowerblue]

symbol - Optional. Change the symbol of the scatter plot points. See list of possible symbols at SAS product documentation (markerattrs symbol). [Default= circle]

size - Optional. Change the size (in pixels) of the plot points. [Default= 6]

loess - Optional. To overlay a loess fit to the scatter plot for comparison. Logical, {TRUE,FALSE}. [Default= FALSE].

cl - Optional. Option to create confidence limits for the regression line. {CLM, CLI}. [Default= CLI]

degree - Optional. Specifies linear or quadratic fit. For linear, degree=1, for quadratic, degree=2. [Default= 1]

alpha - Optional. If ellipse is overlayed, specifies value of alpha for predictive bands. [Default= 0.05]

dateColumn - Optional. Specifies the date column for returns in the data set. [Default= Date]

**Author**

Dominic Pazzula, Carter Johnston

**Examples**

%let dir=C:\SVN\SAS\_Perf\_Anly;

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***Chart\_Regression***(prices, xvar=IBM, yvar=GE);

/\*Or\*/

/\*%Chart\_Regression(prices, xvar=IBM, yvar=GE, title= My simple Regression, loess=TRUE, degree=2, dateColumn=DATE);\*/

**[Chart\_RelativePerformance](#ChartRelativePerformance_TOC)** Chart of relative performance

**Description**

Compare the relative performance of an asset to a benchmark and create the chart.

**Usage**

%Chart\_RelativePerformance(returns)

%Chart\_RelativePerformance(returns, Rf=, title=, method=, dateColumn=)

**Arguments**

returns - Required. Data Set containing returns with option to include risk free rate variable.

Rf - Optional. The value or variable representing the risk free rate of return. [Default=0]

title - Optional. Title for chart. [Default= Relative Performance Against &Rf]

method - Optional. Specifies either DISCRETE or LOG chaining method {DISCRETE, LOG}. [Default=DISCRETE]

dateColumn - Optional. Specifies the date column for returns in the data set. [Default= Date]

**Author**

Dominic Pazzula, Carter Johnston

**Examples**

%let dir=C:\SVN\SAS\_Perf\_Anly;

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***Chart\_RelativePerformance***(prices);

/\*Or\*/

/\*%Chart\_RelativePerformance(prices, Rf=0.0005, title= My Relative Performance, method=LOG, dateColumn=DATE);\*/

**[Chart\_Scatter](#ChartScatter_TOC)** a scatter chart

**Description**

Create a chart of scatter that shows the simple correlation between two asset returns with the option to overlay a regression line.

**Usage**

%Chart\_Scatter(returns, xvar=, yvar=,)

%Chart\_Scatter(returns, xvar=, yvar=, title=, grid=, regLine=, dateColumn=)

**Arguments**

returns - Required. Data Set containing returns.

xvar - Required. Specifies the variable or asset to be plotted on the x-axis.

yvar - Required. Specifies the variable or asset to be plotted on the y-axis.

title - Optional. Title for Scatter Plot. [Default= xvar versus yvar Scatter]

grid - Optional. Overlay a grid aligned with the points on the x and y axis. {TRUE, FALSE} [Default= FALSE]

transparency - Optional. Specifies the level of transparency for data symbols. [Default= 0.35]

color - Optional. To change the color of the scatter plot points [Default= cornflowerblue]

symbol - Optional. To change the symbol of the scatter plot points. See list of possible symbols at SAS product documentation (markerattrs symbol). [Default= circle]

size - Optional. To change the size (in pixels) of the plot points. [Default= 6]

regLine - Optional. Overlay a regression line on the scatter plot. {TRUE, FALSE}. [Default= FALSE]

cl - Optional. If regLine= TRUE, option to create confidence limits for the regression line. {CLM, CLI}. [Default= CLI]

degree - Optional. If regLine= TRUE, specifies linear or quadratic fit. For linear, degree=1, for quadratic, degree=2. [Default= 1]

ellipse - Optional. Add a predictive ellipse to scatter plots. {True, False}. [Default= FALSE]

EllipseType - Optional. If ellipse is overlayed, specifies type. {mean, predicted}. [Default= predicted]

alpha - Optional. If ellipse is overlayed, specifies value of alpha for predictive bands. [Default= 0.05]

dateColumn - Optional. Specifies the date column for returns in the data set. [Default= Date]

**Author**

Dominic Pazzula, Carter Johnston

**Examples**

%let dir=C:\SVN\SAS\_Perf\_Anly;

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***Chart\_Scatter***(prices, xvar=IBM, yvar=GE);

/\*Or\*/

/\*%Chart\_Scatter(prices, xvar=IBM, yvar=GE, title= My Scatter Chart, grid=TRUE, color=red, regLine=True, cl=CLM, ellipse=TRUE, dateColumn=DATE);\*/

**[CoMoments](#CoMoments_TOC)**  Macro for calculating the co-moments of financial time series

**Description**

Calculates the co-skewness and co-kurtosis as the skewness and kurtosis of two assets with reference to one another. This data is input into two separate matrices. CoMoments is an internal macro used in table\_HigherMoments, but can be exposed if the user wishes to see the output directly.

**Details**

The individual elements of the co-skewness matrix can be obtained as:

.

Similarly, the individual elements of the co-kurtosis matrix can be obtained as:

.

**Usage**

%CoMoments(Returns);

%CoMoments(Returns,dateColumn=, outCoSkew=, outCoKurt=);

**Arguments**

Returns - required. Names the data set containing time series data of asset returns.

dateColumn- specifies the date column in the data set. [Default= Date]

outCoSkew- output co-skewness matrix. [Default= M3]

outCoKurt- output co-kurtosis matrix. [Default= M4]

**Author**

Dominic Pazzula, Carter Johnston

**See Also**

[Return\_Centered](#name_Return_Centered), [BetaCoMoments](#name_BetaCo)

**Examples**

%let dir=C:\SVN\SAS\_Perf\_Anly;

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***Return\_Calculate***(prices);

%***CoMoments***(prices); /\*Or\*/

/\*%CoMoments (prices, dateColumn= Date, outCoSkew= M3, outCoKurt= M4);\*/

**[create\_hash](#CreateHash_TOC)** Declare a hash object

**Description**

A helper macro to declare a hash object to be used in DATA STEP. See SAS reference on “DECLARE statement, Hash and Hash Iterator Objects”.

**Usage**

%create\_hash(name=, key=, data\_vars=, dataset=)

**Arguments**

name - Required. Specifies the name of the hash object. {ie. name = my\_hash}

key - Required. Lookup keys to initialize hash object. {ie. key = id}

data\_vars - Required. Specifies the data variables which is to be munipulated. {ie. data\_vars = salary}

dataset - Required. Name of the data set. {ie. dataset = "my\_data\_set"}

**Author**

Dominic Pazzula, Carter Johnston

**Example**

**data** group;

input sex $ groups $;

datalines;

M A

F B

;

**run**;

**data** all(drop=rc);

length groups $1;

set sashelp.class;

if \_n\_ = 1 then do;

%***create\_hash***(name=my\_hash,key=sex,data\_vars=groups,dataset=”group”);

end;

if my\_hash.find() = 0;

**run**;

**[download\_ff3](#Downloadff3_TOC)** download ff3 factors

**Description**

A helper macro that downloads up-to-date daily Fama French 3 factors from online source. The data starts from Jul 1st, 1926.

**Usage**

%download\_ff3(outData=);

**Arguments**

outData- output data set with ff3 factors. [Default= ff3]

**Author**

Dominic Pazzula, Carter Johnston

**Example**

%download\_ff3(outData=my\_ff3);

**[Drawdowns](#Drawdowns_TOC)** calculate drawdowns

**Description**

Calculate the decline of asset return from historical peaks for periodical time series. Expressed as percentages of losses from peak returns.

**Details**

A drawdown is the decline of an investment since its most recent peak price. If the return of the investment is positive, the drawdown is zero. Therefore, the calculation of drawdown is expressed as:

**Usage**

%Drawdowns(returns);

%Drawdowns(returns, method=, dateColumn=, outData=);

**Arguments**

returns - Required. Data Set containing returns.

method - Optional. Specifies either DISCRETE or LOG chaining method {DISCRETE, LOG}. [Default=DISCRETE]

dateColumn - Optional. Date column in Data Set. [Default=DATE]

outData - Optional. Output Data Set with drawdowns. [Default="drawdownPeak"]

**Author**

Qiyuan Yang

**Example**

%let dir=C:\SVN\SAS\_Perf\_Anly;

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***Drawdowns***(prices);

/\*Or\*/

/\*%Drawdowns(prices, method=LOG, dateColumn=DATE, outData=MyDrawdownPeak);\*/

**[Drawdown\_Deviation](#DrawdownDev_TOC)** calculate standard deviation of drawdowns

**Description**

Given a return data set, this macro calculates drawdowns first and then output the standard deviation of drawdowns.

**Details**

The standard deviation of drawdown is calculated as:

Where d is total number of drawdowns, Dt is the tth drawdown, and n is the total number of observations.

**Usage**

%Drawdown\_Deviation(returns)

%Drawdown\_Deviation(returns, method=, dateColumn=, outData)

**Arguments**

returns - Required. Data Set containing returns.

method - Optional. Specifies either DISCRETE or LOG chaining method {DISCRETE, LOG}. [Default=DISCRETE]

dateColumn - Optional. Date column in Data Set. [Default=DATE]

outData - Optional. Output Data Set with drawdown deviation. [Default="DD\_dev "]

**Author**

Qiyuan Yang

**Examples**

%let dir=C:\SVN\SAS\_Perf\_Anly;

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***Drawdown\_Deviation***(prices);

/\*Or\*/

/\*%Drawdown\_Deviation(prices, method=LOG, dateColumn=DATE, outData=MyDDdev);\*/

**[Fama\_Beta](#famabeta_TOC)** Fama beta of the return distribution

**Description**

Fama beta is a beta used to calculate the loss of diversification. It is made so that the systematic risk is equivalent to the total portfolio risk.

**Details**

Fama beta is calculated as:

where is the portfolio standard deviation and is the market risk.

**Usage**

%Fama\_beta(Returns, BM=);

%Fama\_beta(Returns, BM=, dateColumn=, outData=);

**Arguments**

Returns - required. Names the data set containing time series data of asset returns.

BM- required. Names the benchmark asset or index in the data set.

Scale- optional. Number of periods in a year {any positive integer, ie daily scale= 252, monthly scale= 12, quarterly scale= 4}. [Defaul=1]

VARDEF - optional. Specify the variance divisor, DF, degree of freedom, n-1; N, number of observations, n. {N, DF} Default= DF.

dateColumn- specifies the date column in the data set. [Default= Date]

outData- names the output data set with values of Fama beta. [Default= fama\_beta]

**Author**

Dominic Pazzula, Carter Johnston

**Examples**

%let dir=C:\SVN\SAS\_Perf\_Anly;

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***Return\_Calculate***(prices);

%***Fama\_beta***(prices, BM= SPY);

/\*Or\*/

/\*%Fama\_beta(prices,BM= SPY, dateColumn= Date, outData= fama\_beta);\*/

**[Find\_Drawdowns](#FindDrawdowns_TOC)** find starting and ending points of drawdown, as well as length of the interval

**Description**

This macro finds the points in a data set which indicate the starting point, ending point, length, and depth of drawdowns.

**Details**

In the output, variable ‘trough’ indicates the point where the max drawdown of the particular interval occurs. Variables ‘peaktotrough’ and ‘recovery’ indicate the number of points before and after ‘trough’.

**Usage**

%Find\_Drawdowns(returns, asset=);

%Find\_Drawdowns(returns, asset=, method=, dateColumn=, SortDrawdown=, outData=);

**Arguments**

returns - Required. Data Set containing returns.

asset - Required. Name of the variable to find drawdown interval for.

method - Optional. Specifies either DISCRETE or LOG chaining method. {DISCRETE, LOG}. [Default=DISCRETE]

dateColumn - Optional. Date column in Data Set. [Default=DATE]

SortDrawdown - Optional. Specifies whether to sort the data on drawdown depth. {TRUE, FALSE} [Default=FALSE]

outData - Optional. Output Data Set with drawdowns. [Default="FindDrawdowns"]

**Author**

Ruicheng Ma

**See Also**

[Drawdowns](#name_Drawdowns), [max\_drawdown](#MaxDrawdown_TOC), [Sharpe\_Ratio](#name_SharpeRAtio), [Calmar\_Ratio](#name_CalmarRatio), [Sort\_Drawdowns](#name_SortDrawdowns)

**Example**

%let dir=C:\SVN\SAS\_Perf\_Anly;

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***Find\_Drawdowns***(prices, asset=IBM);

/\*Or\*/

/\*%Find\_Drawdowns(prices, asset=IBM, method=LOG, dateColumn=DATE, SortDrawdown=FALSE, outData=MyFindDrawdowns);\*/

**[Geo\_Mean](#GeoMean_TOC)** calculate the geometric mean of the observation series

**Description**

The geometric mean is a measure of central tendency, using multiplication instead of the traditional addition to summarize data values. Geometric means are useful summaries for highly skewed data. Do not use a geometric mean if there are negative or zero values in the data set.

**Details**

Geo\_Mean is intended as a wrapper function to be used inside other macros that require it. However, the user can call Geo\_Mean if they so choose. The geometric mean is given by one of two formulas:

**Usage**

%Geo\_Mean (Returns, BM=, Rf=, scale=);

%Geo\_Mean(Returns, BM=, Rf=, scale=, dateColumn=, outData=);

**Arguments**

Returns- required. Data Set containing returns.

dateColumn- specifies the date column in the data set. [Default= Date]

outData- output Data Set with of systematic risk. [Default= “\_geoMean”]

**Author**

Dominic Pazzula, Carter Johnston

**Examples**

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***Geo\_Mean***(prices);

/\*Or\*/

/\*%Geo\_Mean(prices, dateColumn= Date, outData= \_geoMean);\*/

**[get\_stocks](#GetStocks_TOC)** download daily stock price and return data

**Description**

A helper macro that downloads daily stock price and return data from yahoo.finance, with the option to keep the price.

**Usage**

%get\_stocks(stocks=, from=, to=);

%get\_stocks(stocks=, from=, to=, keepPrice=, LogReturn=, PriceColumn=, outReturns=)

**Arguments**

stocks - Required. Tickers of the stocks. {ie.stocks=IBM GE}

from - Optional. Starting date (inclusive). {ie. 31DEC2004} [Default = 1 year before today's date]

to - Optional. Ending data (inclusive). {ie. 01JAN2015} [Default = 1 day before today's date]

keepPrice - Optional. Specify whether to keep the price data. {0, 1} [Default = 0]

LogReturn - Optional. Compound or single returns. {0, 1} [Default = 1]

PriceColumn - Optional. Specify the kind of price to be kept. [Default = adj\_close]

outReturns - Optional. Output data set with returns. [Default = returns]

**Author**

Dominic Pazzula, Carter Johnston

**Example**

%let stocks = IBM GE DOW GOOGL SPY;

%***get\_stocks***(&stocks);

/\*Or\*/

/\*%get\_stocks(&stocks, from=31DEC2010, to=01JAN2016, keepPrice=1, outReturns=StockReturn);\*/

**[Information Ratio](#InformationRatio_TOC)** Active Premium over Tracking Error

**Description**

Information Ratio is defined as the Active Premium divided by the Tracking Error

**Details**

The information ratio relates the *degree* to which an investment has beaten the benchmark to the *consistency* with which the investment has beaten the benchmark. William Sharpe recommends the information ratio preferentially to the original Sharpe Ratio.

The information Ratio is given by the following formula:

**Information Ratio**=

where

**Usage**

%InformationRatio(Returns, BM=, scale=);

%InformationRatio(Returns,BM=, scale=, dateColumn=, outData=);

**Arguments**

Returns - required. Names the data set containing time series data of asset returns.

BM- required. Names the benchmark asset or index variable in the data set.

Scale- optional. The number of periods in a year (daily scale= 252, monthly= 12) for which the return is to be annualized. Default= 1.

VARDEF - optional. Specify the variance divisor, DF, degree of freedom, n-1; N, number of observations, n. {N, DF} Default= DF.

dateColumn- specifies the date column in the data set. [Default= Date]

outData- output data set with information ratios. [Default= Info\_Rat]

**Author**

Dominic Pazzula, Carter Johnston

**See Also**

[TrackingError](#name_trackingError)

**Examples**

%let dir=C:\SVN\SAS\_Perf\_Anly;

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***Information\_Ratio***(prices, BM= SPY, scale= **252**);

/\*Or\*/

/\*%Information\_Ratio(prices, BM= SPY, scale= 252, dateColumn= Date, outData= Info\_Rat);\*/

**[Martin\_Ratio](#MartinRatio_TOC)** calculate Martin Ratio

**Description**

Martin Ratio is calculated by taking the difference between asset return and risk free rate and divided by Ulcer Index.

**Details**

The Ulcer Index and Martin Ratio can be calculated as:

**Usage**

%Martin\_Ratio(returns);

%Martin\_Ratio(returns, Rf=, scale=, method=, dateColumn=, outData=);

**Arguments**

returns - Required. Data Set containing returns.

Rf - Optional. the value or variable representing the risk free rate of return. [Default=0]

scale - Optional. Number of periods in a year {any positive integer, ie daily scale= 252, monthly scale= 12, quarterly scale= 4}. [Default=1]

method - Optional. Specifies either DISCRETE or LOG chaining method {DISCRETE, LOG}. [Default=DISCRETE]

dateColumn - Optional. Date column in Data Set. [Default=DATE]

outData - Optional. Output Data Set with martin ratio. [Default="MartinRatio"]

**Author**

Qiyuan Yang

**Example**

%let dir=C:\SVN\SAS\_Perf\_Anly;

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***Martin\_Ratio***(prices);

/\*Or\*/

/\*%Martin\_Ratio(prices, Rf=0, method=LOG, dateColumn=DATE, outData=MyMartinRatio);\*/

**[max\_drawdown](#MaxDrawdown_TOC)** calculate drawdowns

**Description**

Finds the worst drawdown for the whole time period for every asset. The drawdown is calculated based on cumulative return. The ‘invert’ option allows user to output drawdown as either positive or negative numbers to serve different purposes.

**Usage**

%max\_drawdowns(returns);

%max\_drawdowns(returns, method=, invert=, dateColumn=, outData=);

**Arguments**

returns - Required. Data Set containing returns.

method - Optional. Specifies either DISCRETE or LOG chaining method {DISCRETE, LOG}. [Default=DISCRETE]

invert - Optional. Specifies whether to invert the drawdown measure. [Default=TRUE]

dateColumn - Optional. Date column in Data Set. [Default=DATE]

outData - Optional. Output Data Set with maximum drawdowns. [Default="max\_dd"]

**Author**

Qiyuan Yang

**See Also**

[Drawdowns](#name_Drawdowns)

**Example**

%let dir=C:\SVN\SAS\_Perf\_Anly;

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***max\_drawdown***(prices);

/\*Or\*/

/\*%max\_drawdown(prices, method=LOG, invert=FALSE, dateColumn=DATE, outData=MyMaxDrawdowns);\*/

**[Mean\_Abs\_Deviation](#MeanAbsDeviation_TOC)** calculate mean absolute deviation

**Description**

Calculate mean absolute deviation. It is defined as the sum of absolute value of difference between the returns and average return divided by total number.

**Details**

**Usage**

%Mean\_Abs\_Deviation(returns=);

%Mean\_Abs\_Deviation(returns=, dateColumn=, outData=);

**Arguments**

returns - Required. Data Set containing returns.

dateColumn - Optional. Date column in Data Set. [Default=DATE]

outData - Optional. Output Data Set with mean absolute deviation. [Default="mean\_abs\_dev"]

**Author**

Qiyuan Yang

**Example**

%let dir=C:\SVN\SAS\_Perf\_Anly;

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***Mean\_Abs\_Deviation***(prices);

/\*Or\*/

/\*%Mean\_Abs\_Deviation(prices, dateColumn=DATE, outData=MyMeanAbsDev);\*/

**[MSquared](#MSquared_TOC)** M squared of the return distribution

**Description**

M squared is a risk adjusted return useful to judge the size of relative performance between different portfolios. With it you can compare portfolios with different levels of risk.

**Details**

**Usage**

%MSquared (Returns, BM=);

%MSquared(Returns, BM=, Rf=, scale=, method=, dateColumn=, outData=);

**Arguments**

Returns - required. Names the data set containing time series data of asset returns.

BM- required. Names the benchmark asset or index variable in the data set.

Rf- specifies the variable or value of the risk free rate of return.

Scale- optional. The number of periods in a year (daily scale= 252, monthly= 12) for which the return is to be annualized. Default= 1.

Method- option to implement either DISCRETE or LOG chaining when annualizing returns.

VARDEF - optional. Specify the variance divisor, DF, degree of freedom, n-1; N, number of observations, n. {N, DF} Default= DF.

dateColumn- specifies the date column in the data set. [Default= Date]

outData- output data set with M squared values. [Default= MSquared]

**Author**

Dominic Pazzula, Carter Johnston

**Notes**

The authors would like to express that this macro is distinct from the function given by R performance analytics due to disagreements with its implementation. The definition of M squared is the Sharpe Ratio multiplied by the standard deviation of the benchmark plus the risk free rate. When annualizing M squared, the geometric average ought to be calculated as the geometric average of , not the geometric average of . For this reason, results between the macro and its equivalent in R will be different.

**Examples**

%let dir=C:\SVN\SAS\_Perf\_Anly;

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***MSquared***(prices, BM= SPY, scale= **252**);

/\*Or\*/

/\*%MSquared(prices, BM= SPY, Rf= 0.01/252, scale= 252, method= DISCRETE, dateColumn= Date, outData= MSquared);\*/

**[Pain\_Index](#PainIndex_TOC)** calculate Pain Index

**Description**

Pain Index is the mean value of drawdowns. It is notable that to calculate the mean value, the sum of absolute value of drawdowns is divided by the total number of observations, rather than the number of drawdowns.

**Details**

The Pain Index can be calculated as:

Where n denotes the number of observations from the data set, and Dt is the tth drawdown.

**Usage**

%Pain\_Index(returns);

%Pain\_Index(returns, method=, dateColumn=, outData=);

**Arguments**

returns - Required. Data Set containing returns.

method - Optional. Specifies either DISCRETE or LOG chaining method {DISCRETE, LOG}. [Default=DISCRETE]

dateColumn - Optional. Date column in Data Set. [Default=DATE]

outData - Optional. Output Data Set with pain index. [Default="PainIndex"]

**Author**

Qiyuan Yang

**Example**

%let dir=C:\SVN\SAS\_Perf\_Anly;

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***Pain\_Index***(prices);

/\*Or\*/

/\*%Pain\_Index(prices, method=LOG, dateColumn=DATE, outData=MyPainIndex);\*/

**[Pain\_Ratio](#PainRatio_TOC)** calculate Pain Ratio

**Description**

Pain Ratio is calculated by taking the difference between asset return and risk free rate and divided by Pain Index.

**Details**

The Pain Index and Pain Ratio can be calculated as:

**Usage**

%Pain\_Ratio(returns);

%Pain\_Ratio(returns, Rf=, scale=, method=, dateColumn=, outData=);

**Arguments**

returns - Required. Data Set containing returns.

Rf - Optional. the value or variable representing the risk free rate of return. [Default=0]

scale - Optional. Number of periods in a year {any positive integer, ie daily scale= 252, monthly scale= 12, quarterly scale= 4}. [Default=1]

method - Optional. Specifies either DISCRETE or LOG chaining method {DISCRETE, LOG}. [Default=DISCRETE]

dateColumn - Optional. Date column in Data Set. [Default=DATE]

outData - Optional. Output Data Set with pain ratio. [Default="PainRatio"]

**Author**

Qiyuan Yang

**Example**

%let dir=C:\SVN\SAS\_Perf\_Anly;

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***Pain\_Ratio***(prices);

/\*Or\*/

/\*%Pain\_Ratio(prices, Rf=0, scale=252, method=LOG, dateColumn=DATE, outData=MyPainRatio);\*/

**[Prices](#prices_TOC)** Selected price series example data

**Description**

An example price timeseries data set produced by %get\_Stocks which was used to test the majority of the macros in this library, as well as the data set used in examples in this documentation.

**Usage**

%Return\_Calculate(prices);

**Format**

SAS data set

**Examples**

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

[Return\_Annualized](#Return_Annualized_TOC) calculate an annualized return for comparing instruments with different length history

**Description**

Calculate annualized returns for comparison between instruments with different length history.

**Details**

Annualized returns are useful for comparing two assets. Observations are scaled to an annual scale by raising the compound return to the number of periods in a year, and taking the root to the number of total observations:

where scale is the number of periods in a year, and n is the total number of observations in terms of periods.

For simple arithmetic returns, the formula is simply

**Usage**

%Return\_Annualized (Returns, scale=);

%Return\_Annualized(Returns, scale=, method=, dateColumn, outData=);

**Arguments**

Returns - required. Names the data set containing time series data of asset returns.

Scale- required. The number of periods in a year (daily scale= 252, monthly= 12) for which the return is to be annualized.

Method- {DISCRETE, LOG} –option to implement DISCRETE or LOG chaining. [Default= DISCRETE]

dateColumn- specifies the date column in the data set. [Default= Date]

outData- output data set with annualized returns. [Default= annualized\_returns]

**Author**

Dominic Pazzula, Carter Johnston

**Examples**

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***Return\_Annualized***(prices, scale= **252**, method= DISCRETE)

/\*Or\*/

/\*%Return\_Annualized(prices, scale= 252, method= LOG, dateColumn= Date, outData= annualized\_returns);\*/

**[Return\_Calculate](#Return_Calculate_TOC)** calculate simple or compound returns from prices

**Description**

Calculate simple or compound returns from a series of prices. Option to update the table in place or create new output.

**Details**

There are two requirements for Return\_Calculate that should be made clear. First, price data is assumed to be regular. Prices can be for any time scale as long as the data consists of regular observations. Irregular observations require time period scaling to be comparable. Second, if corporate actions, such as a stock-split, dividends, or other adjustments such as time or money weighting are to be taken into account, those calculations must be made separately. This macro assumes fully adjusted close prices as input. The default for this function is to use discrete returns, as most other macros in the library utilize compound chaining by default.

**Usage**

%Return\_Calculate (Prices);

%Return\_Calculate(Prices, method=, dateColumn=, updateInPlace=, outData=);

**Arguments**

Prices- required. Data Set containing prices

Method- {LOG, DISCRETE} – compound or simple returns. [Default= DISCRETE]

dateColumn- specifies the date column in the data set. [Default= Date]

updateInPlace- {True, False} – update the &prices Data Set in place. [Default= TRUE]

outData- output Data Set with returns. Only used if updateInPlace= FALSE [Default= “returns”]

**Author**

Dominic Pazzula

**See Also**

[Return\_Cumulative](#name_Return_Cumulative)

**Examples**

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

/\*Or\*/

/\*%Return\_Calculate(prices, method= DISCRETE, dateColumn= Date, updateInPlace= TRUE, outData= returns);\*/

[Return\_Centered](#Return_Centered_TOC) calculate centered returns

**Description**

Calculate the values of centered returns from a series of returns.

**Details**

The centered return is calculated as return minus the expected return, or mean, of an instrument. It is used in the calculations of higher moments.

**Usage**

%Return\_Centered (Returns);

%Return\_Centered(Returns, dateColumn=, outData=);

**Arguments**

Returns- required. Data Set containing returns

dateColumn- specifies the date column in the data set. [Default= Date]

outData- output Data Set with centered returns. [Default= “centered\_returns”]

**Author**

Dominic Pazzula, Carter Johnston

**See Also**

[CoMoments](#name_CoMoments), [table\_HigherMoments](#name_tableHigherMoments)

**Examples**

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***Return\_Centered***(prices);

/\*Or\*/

/\*%Return\_Centered(prices, dateColumn= Date, outData= centered\_returns);\*/

**[Return\_Cumulative](#Return_Cumulative_TOC)** calculate a compounded cumulative return

**Description**

Calculates the cumulative return over a period of time, producing simple or geometric returns.

**Details**

The product of all individual period returns is

If calculating the cumulative return using the simple arithmetic method, cumulative returns are equivalent to the sum of all returns.

**Usage**

%Return\_Cumulative (Returns);

%Return\_Cumulative(Returns, method=, dateColumn=, outData=);

**Arguments**

Returns- required. Data Set containing returns.

Method- {DISCRETE, LOG} – specifies compound or simple returns. [Default= DISCRETE]

dateColumn- specifies the date column in the data set. [Default= Date]

outData- output Data Set with cumulative returns. [Default= “cumulative\_returns”]

**Author**

Dominic Pazzula, Carter Johnston

**See Also**

[Return\_Annualized](#name_Return_Annualized)

**Examples**

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***Return\_Cumulative***(prices);

/\*Or\*/

/\*%Return\_Cumulative(prices, method= DISCRETE, dateColumn= Date, outData= cumulative\_returns);\*/

**[Return\_Excess](#Return_Excess_TOC)** Calculate the returns of an asset in excess of the given risk free rate

**Description**

Calculates the returns of an asset in excess of the given “risk free rate” for the period.

**Details**

Ideally, the risk free rate will be for each period in which returns data is available, but a single average risk free rate for the period will work with this macro. While the Rf parameter is named after the risk free rate, any timeseries is allowed. A common alteration would be to find the excess returns over a benchmark asset or index.

**Usage**

%Return\_Excess (Returns, Rf=);

%Return\_Excess(Returns, Rf=, dateColumn=, outData=);

**Arguments**

Returns- required. Data Set containing returns.

Rf- required. A value or vector representing the risk free rate of return.

dateColumn- specifies the date column in the data set. [Default= Date]

outData- output Data Set with risk premium. [Default= “risk\_premium”]

**Author**

Dominic Pazzula, Carter Johnston

**See Also**

[Return\_Calculate](#name_Return_Calculate)

**Examples**

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***Return\_Excess***(prices, Rf= 0.01/252);

/\*Or\*/

/\*%Return\_Excess(prices, Rf= SPY, dateColumn= Date, outData= risk\_premium);\*/

**[Sharpe\_Ratio](#Sharpe_Ratio_TOC)** calculate a traditional Sharpe Ratio of return over standard deviation.

**Description**

The Sharpe ratio is simply the return per unit of risk (represented by variability). The original Sharpe ratio uses standard deviation of returns as unit of risk.

**Details**

The Sharpe Ratio, in its original form, is the expected excess return over the standard deviation of excess return. The higher the Sharpe ratio, the better the combined performance of “risk” and return. It is worth mentioning that William Sharpe, the originator of the Sharpe ratio, now recommends the Information ratio preferentially to the original Sharpe Ratio.

It should be noted that the equivalent function in R performance analytics allows the user to calculate modified Sharpe Ratios with Value at Risk (VaR) or Expected Shortfall (ES) instead of Standard Deviation. We are working on adding these options to %Sharpe\_Ratio and will hopefully have this updated soon.

**Usage**

%Sharpe\_Ratio (Returns, Rf=);

%Sharpe\_Ratio(Returns, Rf=, dateColumn=, outData=);

**Arguments**

Returns- required. Data Set containing returns.

Rf- required. A value or vector representing the risk free rate of return.

VARDEF - optional. Specify the variance divisor, DF, degree of freedom, n-1; N, number of observations, n. {N, DF} Default= DF.

dateColumn- specifies the date column in the data set. [Default= Date]

outData- output Data Set with Sharpe ratios. [Default= “SharpeRatio”]

**Author**

Dominic Pazzula, Carter Johnston

**See Also**

[Information\_Ratio](#name_InformationRatio)

**Examples**

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***Sharpe\_Ratio***(prices, Rf= 0.01/252);

/\*Or\*/

/\*%Sharpe\_Ratio(prices, Rf= SPY, dateColumn= Date, outData= SharpeRatio);\*/

**[SharpeRatio\_Annualized](#SharpeRatioAnnualized_TOC)** calculate annualized Sharpe ratio

**Description**

The Sharpe ratio is a risk-adjusted measure of return that uses standard deviation to represent risk. The annualized Sharpe ratio is calculated using both annualized returns and annualized standard deviation to measure return.

**Details**

The Sharpe ratio is simply the return per unit of risk (represented by variance). The higher the Sharpe ratio, the better the combined performance of “risk” and return. This macro annualizes this ratio based on the scale parameter which specifies the number of periods for which there is data in one year.

Using an annualized Sharpe ratio is useful for comparison of multiple return streams where length of holdings are not necessarily equal.

**Usage**

%SharpeRatio\_Annualized (Returns, Rf=, scale=);

%SharpeRatio\_Annualized(Returns, Rf=, scale=, method=, dateColumn=, outData=);

**Arguments**

Returns- required. Data Set containing returns.

Rf- required. A value or vector representing the risk free rate of return.

Scale- required. Specifies the number of periods in one year. [Daily= 252, Monthly= 12, Quarterly= 4]

Method- option for DISCRETE or LOG chaining. {DISCRETE, LOG} [Default= DISCRETE]

VARDEF - optional. Specify the variance divisor, DF, degree of freedom, n-1; N, number of observations, n. {N, DF} Default= DF.

dateColumn- specifies the date column in the data set. [Default= Date]

outData- output Data Set with annualized Sharpe ratios. [Default= “Annualized\_SharpeRatio”]

**Author**

Dominic Pazzula, Carter Johnston

**See Also**

[Sharpe\_Ratio](#name_SharpeRAtio)

**Examples**

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***SharpeRatio\_Annualized***(prices, Rf= 0.01/252, scale= 252);

/\*Or\*/

/\*%SharpeRatio\_Annualized(prices, Rf= SPY, scale= 252, method= DISCRETE, dateColumn= Date, outData= SharpeRatio);\*/

**[simple\_normalize](#SimpleNormalize_TOC)** assign weight to each value

**Description**

Calculate the weight of each observation under a variable with the option of choosing a total weight. The original values from input dataset will be replaced by the weight.

**Usage**

%simple\_normalize(data=, var=, sum=);

**Arguments**

data - Required. Data set containing required variable.

var - Required. Name of the variable to be calculated.

sum - Optional. The total weight of the variable. [Default = 1]

**Author**

Dominic Pazzula, Carter Johnston

**Example**

**data** airlines;

set sashelp.airline;

**run**;

%***simple\_normalize***(airlines, var=air, sum=1);

**[simple\_normalize\_by](#SimpleNormalizeBy_TOC)** assign weight to each value by group

**Description**

Calculate the weight of each observation under a variable by groups.

**Usage**

%simple\_normalize\_by(data=, var=, by=);

**Arguments**

data - Required. Data set that contains the data of interest. {ie. data = my\_data\_set}

var - Required. Name of the variable to be manipulated.

by - Required. Speficies the variable that divides data into subgroups.

**Author**

Dominic Pazzula, Carter Johnston

**Example**

**data** electric;

set sashelp.electric;

keep revenue year;

format revenue best.;

**run**;

%***simple\_normalize\_by***(electric,var=revenue,by=year);

**[Sort\_Drawdowns](#SortDrawdowns_TOC)** Sort data based on drawdown return

**Description**

Given the drawdowns data from Find\_Drawdowns macro, sort return variable from lowest to highest.

**Usage**

%Sort\_Drawdowns(returns, asset=);

%Sort\_Drawdowns(returns, asset=, method=, dateColumn=, outData=);

**Arguments**

returns - Required. Data Set containing returns.

asset - Required. Name of the variable to find drawdown interval for.

method - Optional. Specifies either DISCRETE or LOG chaining method {DISCRETE, LOG}. [Default=DISCRETE]

dateColumn - Optional. Date column in Data Set. Default=DATE

outData - Optional. Output Data Set with drawdowns. Default="SortDrawdown".

**Author**

Ruicheng Ma

**See Also**

[Drawdowns](#name_Drawdowns), [max\_drawdown](#name_MaxDrawdown), [Find\_Drawdowns](#name_FindDrawdowns), [Table\_Drawdowns](#name_TableDrawdowns)

**Example**

%let dir=C:\SVN\SAS\_Perf\_Anly;

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***Sort\_Drawdowns***(prices, asset=IBM);

/\*Or\*/

/\*%Sort\_Drawdowns(prices, asset=IBM, method=LOG, dateColumn=DATE, outData=MySortDrawdowns);\*/

**[Specific\_Risk](#SpecificRisk_TOC) specific risk of the return distribution**

**Description**

Specific risk is the standard deviation of the error term in the regression equation. Specific risk is not the same as market risk. Market risk is the standard deviation of the benchmark asset or index.

**Details**

Specific risk can be thought of as the standard deviation of the regression epsilon, or it can be thought of as the square root of total risk squared minus systematic risk squared.

**Usage**

%Specific\_Risk (Returns, BM=, Rf=, scale=);

%Specific\_Risk(Returns, BM=, Rf=, scale=, dateColumn=, outData=);

**Arguments**

Returns- required. Data Set containing returns.

BM- required. Names variable containing returns of benchmark asset from the returns data set.

Rf- required. A value or vector representing the risk free rate of return.

Scale- required. Specifies the number of periods in one year. [Daily= 252, Monthly= 12, Quarterly= 4]

VARDEF - optional. Specify the variance divisor, DF, degree of freedom, n-1; N, number of observations, n. {N, DF} Default= DF.

dateColumn- specifies the date column in the data set. [Default= Date]

outData- output Data Set with specific risk values. [Default= “Risk\_Specific”]

**Author**

Dominic Pazzula, Carter Johnston

**See Also**

[Systematic\_Risk](#name_SystematicRisk), [table\_SpecificRisk](#name_tableSpecificRisk)

**Examples**

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***Specific\_Risk***(prices, BM= SPY, Rf= 0.01/252, scale= 252);

/\*Or\*/

/\*%Specific\_Risk(prices, BM= SPY, Rf= IBM, scale= 252, dateColumn= Date, outData= Risk\_specific);\*/

**[Standard\_Deviation](#StandardDeviation_TOC) calculates standard deviation for univariate and multivariate series**

**Description**

Calculates the standard deviation, or “risk”, for univariate and multivariate series. This macro contains the option to annualize the standard deviation given the extra parameter, scale.

**Details**

The standard deviation is calculated using the following formula:

Where = return observation of asset, = the mean return of the asset, and n= number of observations.

The annualized standard deviation is calculated as:

**Usage**

%Standard\_Deviation (Returns);

%Standard\_Deviation(Returns, annualized=, scale=, dateColumn=, outData=);

**Arguments**

Returns- required. Data Set containing returns.

annualized - Optional. Option to annualize the standard deviation. {TRUE, FALSE} [Default= FALSE]

scale - Optional. Number of periods in a year {any positive integer, ie daily scale= 252, monthly scale= 12, quarterly scale= 4}. [Default= 1]

VARDEF - optional. Specify the variance divisor, DF, degree of freedom, n-1; N, number of observations, n. {N, DF} Default= DF.

dateColumn- specifies the date column in the data set. [Default= Date]

outData- output Data Set with standard deviations. [Default= “StdDev”]

**Author**

Dominic Pazzula, Carter Johnston

**See Also**

[StdDev\_Annualized](#name_StdDevAnnualized)

**Examples**

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***Standard\_Deviation***(prices);

/\*Or\*/

/\*%Standard\_Deviation(prices, dateColumn= Date, outData= StdDev);\*/

[StdDev\_Annualized](#StdDevAnnualized_TOC) calculate a multiperiod or annualized standard deviation

**Description**

Calculate the standard deviation of a set of observations and estimate the annualized standard deviation based on the number of periods in a year.

**Details**

To normalize standard deviation across multiple periods, we multiply by the square root of the number of periods we wish to calculate over. To annualize standard deviation, we multiply by the square root of the number of periods per year.

**Usage**

%StdDev\_Annualized (Returns, scale=);

%StdDev\_Annualized(Returns, scale=, dateColumn=, outData=);

**Arguments**

Returns- required. Data Set containing returns.

scale- If annualized= TRUE, then scale specifies the number of periods in one year. [Daily= 252, Monthly= 12, Quarterly= 4]

annualized - optional. Annualize the standard deviation. {TRUE,FALSE} [Default= FALSE]

VARDEF - optional. Specify the variance divisor, DF, degree of freedom, n-1; N, number of observations, n. {N, DF} Default= DF.

dateColumn- specifies the date column in the data set. [Default= Date]

outData- output Data Set with of systematic risk. [Default= “annualized\_StdDev”]

**Author**

Dominic Pazzula, Carter Johnston

**See Also**

[Standard\_Deviation](#name_StandardDeviation)

**Examples**

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***StdDev\_Annualized***(prices, scale=252);

/\*Or\*/

/\*%StdDev\_Annualized(prices, scale= 252, dateColumn= Date, outData= annualized\_StdDev);\*/

**[Sterling\_Ratio](#SterlingRatio_TOC)** calculate Sterling Ratio

**Description**

Sterling Ratio measures an asset’s return against its drawdown risk like Calmar Ratio. The difference is Sterling Ratio adds an excess risk measure to the drawdown in the denominator.

**Details**

Sterling Ratio is calculated as follows.

**Usage**

%Sterling\_Ratio(returns);

%Sterling\_Ratio(returns, scale=, method=, excess=, dateColumn=, outData=);

**Arguments**

returns - Required. Data Set containing returns.

scale - Optional. Number of periods in a year {any positive integer, ie daily scale= 252, monthly scale= 12, quarterly scale= 4}. [Default=1]

method - Optional. Specifies either DISCRETE or LOG chaining method {DISCRETE, LOG}. [Default=DISCRETE]

excess - Optional. The yield of risk-free investment compared by any investment with a return stream. [Default=0.1]

dateColumn - Optional. Date column in Data Set. [Default=DATE]

outData - Optional. Output Data Set with Sterling ratios. [Default="SterlingRatio"]

**Author**

Qiyuan Yang

**See Also**

[Drawdowns](#name_Drawdowns), [max\_drawdown](#name_MaxDrawdown), [Sharpe\_Ratio](#name_SharpeRAtio), [Calmar\_Ratio](#name_CalmarRatio)

**Example**

%let dir=C:\SVN\SAS\_Perf\_Anly;

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***Sterling\_Ratio***(prices);

/\*Or\*/

/\*%Sterling\_Ratio(prices, scale=252, method=LOG, excess=0.2, dateColumn=DATE, outData=MySterlingRatio);\*/

**[Systematic\_Risk](#SystematicRisk_TOC)** Systematic risk of the return distribution

**Description**

Systematic risk is defined as the product of beta and market risk. Market risk is the standard deviation of the benchmark asset or index. The systematic risk is then annualized for comparison purposes.

**Details**

Where is systematic risk, is the regression beta, and is the market risk.

**Usage**

%Systematic\_Risk (Returns, BM=, Rf=, scale=);

%Systematic\_Risk(Returns, BM=, Rf=, scale=, dateColumn=, outData=);

**Arguments**

Returns- required. Data Set containing returns.

BM- required. Names variable containing returns of benchmark asset from returns data set.

Rf- required. Either a value or vector representing the risk free rate of return.

Scale- If annualized= TRUE, then scale specifies the number of periods in one year. [Daily= 252, Monthly= 12, Quarterly= 4]

VARDEF - optional. Specify the variance divisor, DF, degree of freedom, n-1; N, number of observations, n. {N, DF} Default= DF.

dateColumn- specifies the date column in the data set. [Default= Date]

outData- output Data Set with of systematic risk. [Default= “Risk\_systematic”]

**Author**

Dominic Pazzula, Carter Johnston

**See Also**

[Specific\_Risk](#name_SpecificRisk), [table\_SpecificRisk](#name_tableSpecificRisk)

**Examples**

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***Systematic\_Risk***(prices, BM= SPY, Rf= 0.01/252, scale=252);

/\*Or\*/

/\*%Systematic\_Risk(prices, BM= SPY, Rf= IBM, scale= 252, dateColumn= Date, outData= Risk\_systematic);\*/

**[Table\_Annualized\_Returns](#tableAnnualizedReturns_TOC)** Annualized returns summary: statistics and stylized facts

**Description**

This macro returns a table containing annualized return, annualized standard deviation, and annualized Sharpe ratio.

**Usage**

%table\_Annualized\_Returns (Returns, Rf=, scale=);

%table\_Annualized\_Returns(Returns, Rf=, scale=, method=, dateColumn=, outData=, printTable=);

**Arguments**

Returns- required. Data Set containing returns.

Rf- required. Either a value or vector representing the risk free rate of return.

scale- If annualized= TRUE, then scale specifies the number of periods in one year. [Daily= 252, Monthly= 12, Quarterly= 4]

method- option to compute returns using DISCRETE or LOG chaining. {DISCRETE, LOG} [Default= DISCRETE]

VARDEF - optional. Specify the variance divisor, DF, degree of freedom, n-1; N, number of observations, n. {N, DF} Default= DF.

dateColumn- specifies the date column in the data set. [Default= Date]

outData- output Data Set with annualized returns statistics. [Default= “annualized\_table”]

printTable- option to print the output data set. {PRINT, NOPRINT} [Default= NOPRINT]

**Author**

Dominic Pazzula, Carter Johnston

**See Also**

[Return\_Annualized](#name_Return_Annualized), [Standard\_Deviation](#name_StandardDeviation), [SharpeRatio\_Annualized](#name_SharpeRatioAnnualized)

**Examples**

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***table\_Annualized\_Returns(***prices, Rf= 0.01/252, scale=252);

/\*Or\*/

/\*%table\_Annualized\_Returns(prices, Rf= SPY, scale= 252,method= DISCRETE, dateColumn= DATE, outData= annualized\_table, printTable= PRINT);

**[Table\_AutoCorrelation](#tableAutoCorrelations_TOC)** table for calculating autocorrelation coefficients and significance

**Description**

Produces a table consisting of autocorrelation coefficients *ρ* and corresponding significance levels as measured by the Ljung-Box test Q-statistic for each asset.

More information on the Ljung-Box test and the accompanying Q-test can be found here:

<https://en.wikipedia.org/wiki/Ljung%E2%80%93Box_test>

**Usage**

%Table\_AutoCorrelation (Returns, Rf=, scale=);

%Table\_AutoCorrelation(Returns, Rf=, scale=, method=, annualized=, dateColumn=, outData=, printTable=);

**Arguments**

Returns- required. Data Set containing returns.

Nlag- required. Specifies the number of lags to perform autocorrelations.

dateColumn- specifies the date column in the data set. [Default= Date]

outData- output Data Set with annualized returns statistics. [Default= “AutoCorrelations”]

printTable- Optional. Option to print table. {PRINT, NOPRINT}. Default= [NOPRINT]

**Author**

Dominic Pazzula, Carter Johnston

**Examples**

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***table\_AutoCorrelation(***prices, nlag= 6);

/\*Or\*/

/\*%table\_AutoCorrelation(prices, nlag= 6, dateColumn= DATE, outData=AutoCorrelations, printTable= PRINT);

**[Table\_CalendarReturns](#tableCalendarReturns_TOC)** Monthly and calendar year return table

**Description**

Returns a table of returns formatted with years in rows, months in columns, and a total return in the last column.

**Details**

To mimic results from R performance analytics, the user should specify one asset by name to be printed in the parameters of the macro. If no asset name is found, the table will show results for all assets in order by asset name.

**Usage**

%table\_CalendarReturns(Returns);

%table\_CalendarReturns(Returns, method=, dateColumn=, outData=, printTable=, name=);

**Arguments**

Returns- required. Data set containing returns.

Method- option to compute returns using discrete or log chaining. {DISCRETE, LOG} [Default= DISCRETE]

dateColumn- specifies the date column in the data set. [Default= DATE]

outData- output Data Set with annualized returns statistics. [Default= “Calendar\_Returns”]

printTable- option to print output data set. {PRINT, NOPRINT} [Default= NOPRINT]

name- specifies a single variable or asset name to print if printTable= PRINT.

**Author**

Dominic Pazzula, Carter Johnston

**Examples**

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***table\_CalendarReturns***(prices);

/\*Or\*/

/\*%table\_CalendarReturns(prices, method= DISCRETE, dateColumn= DATE, outData= Calendar\_Returns, printTable= PRINT, name= IBM);

**[Table\_CAPM](#table_CAPM_TOC)** table containing a series of measures of asset excess returns against a benchmark

**Description**

The excess return of assets are used to find a series of measures against a benchmark, based on CAPM model. The market benchmark comes from the input data set and its variable name needs to be specified.

**Usage**

%table\_CAPM(returns=, BM=);

%table\_CAPM(returns=, BM=, scale=, digits=, dateColumn=, outData=, printTable=);

**Arguments**

returns - Required. Data set containing returns of the portfolio.

BM - Required. Specifies the variable name of benchmark asset or index in the returns data set.

Rf - Optional. The value or variable representing the risk free rate of return. [Default=0]

scale - Optional. Number of periods in a year. {daily=252, monthly=12, quarterly=4, yearly=1} [Default=1]

digits - Optional. Specifies number of digits displayed in the output.

VARDEF - Optional. Specifies the variance divisor, DF, degree of freedom, n-1; N, number of observations, n. {N, DF} Default= DF.

dateColumn - Optional. Date column in Data Set. Default=DATE

outData - Optional. Specifies name of output Data Set of correlations. [Default="Correlations"]

printTable - Optional. Option to print output table. {PRINT, NOPRINT} [Default= NOPRINT

**Author**

Qiyuan Yang

**See Also**

CAPM\_alphta\_beta, Bull\_Bear\_beta, TrackingError, Information\_Ratio, Treynor\_Ratio.

**Example**

%let dir=C:\SVN\SAS\_Perf\_Anly;

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***table\_CAPM***(prices, BM=IBM);

/\*Or\*/

/\*%table\_CAPM(prices, BM=IBM, scale=4, outData=MyCAPMTable, dateColumn=DATE, printTable=PRINT);\*/

[Table\_Correlation](#tableCorrelations_TOC) calculate correlations of multiple assets

**Description**

Calculates the correlation, significance, and confidence intervals for correlation estimates with an asset or benchmark index.

**Usage**

%table\_Correlation(Returns, returnsCompare=);

%table\_Correlation(Returns, returnsCompare=, dateColumn=, outData=, printTable=);

**Arguments**

Returns- required. Data Set containing returns.

ReturnsCompare- required. Asset or benchmark index to calculate correlations with.

dateColumn- specifies the date column in the data set. [Default= Date]

outData- output Data Set with annualized returns statistics. [Default= “Correlations”]

printTable- option to print table. {PRINT, NOPRINT}, Default= [NOPRINT].

**Author**

Dominic Pazzula, Carter Johnston

**Examples**

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***table\_Correlation(***prices, returnsCompare= GE);

/\*Or\*/

/\*%table\_Correlation(prices,returnsCompare= SPY, dateColumn= DATE, outData=Correlations, printTable= PRINT);

**[Table\_Distributions](#Table_Distributions_TOC)** Distribtuions summary: statistics and stylized facts

**Description**

Table containing, standard deviation, skewness, sample standard deviation, kurtosis, excess kurtosis, sample skewness, and sample excess kurtosis.

**Usage**

%table\_Distributions(Returns);

%table\_Distributions(Returns, dateColumn=, outData=, digits=, scale=, printTable=);

**Arguments**

Returns- required. Data Set containing returns.

dateColumn- specifies the date column in the data set. [Default= Date]

outData- output Data Set with distribution statistics. [Default= “Correlations”]

digits- specifies the amount of digits to display in output. [Default= 4].

Scale- Denotes the scale for which to annualize the standard deviation. [Default= 1]

VARDEF- Optional. Specify the variance divisor, DF, degree of freedom, n-1; N, number of observations, n. {N, DF} Default= DF.

printTable- option to print table. {PRINT, NOPRINT}, Default= [NOPRINT].

**Author**

Dominic Pazzula, Carter Johnston

**Examples**

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***table\_Distributions(***prices);

/\*Or\*/

/\*%table\_Distributions(prices, dateColumn= DATE, outData=distribution\_table, digits= 8, scale= 252, printTable= PRINT);

**[Table\_Drawdowns](#tableDrawdowns_TOC)** list a table of worst drawdowns

**Description**

Create a table with user specified number of rows to display the statistics of worst drawdowns.

**Usage**

%Table\_Drawdowns(returns, asset=);

%Table\_Drawdowns(returns, asset=, TOP=, method=, digits=, dateColumn=, outData=, printTable=);

**Arguments**

returns - Required. Data Set containing returns.

asset - Required. Name of the variable to find drawdown interval for.

TOP - Optional. The number of the drawdowns with worst depth to include. If the given number is larger than the number of total observations, all drawdowns will be displayed. [Default=5]

method - Optional. Specifies either DISCRETE or LOG chaining method {DISCRETE, LOG}. [Default=DISCRETE]

digits - Optional. Specifies number of digits displayed in the output.

dateColumn - Optional. Date column in Data Set. [Default=DATE]

outData - Optional. Output Data Set with drawdowns. [Default="TableDrawdowns"]

printTable - Optional. Option to print table. {PRINT, NOPRINT} [Default= NOPRINT]

**Author**

Ruicheng Ma

**See Also**

[Drawdowns](#name_Drawdowns), [max\_drawdown](#name_MaxDrawdown), [Find\_Drawdowns](#name_FindDrawdowns), [Sort\_Drawdowns](#name_SortDrawdowns)

**Example**

%let dir=C:\SVN\SAS\_Perf\_Anly;

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***Table\_Drawdowns***(prices, asset=IBM, TOP=5);

/\*Or\*/

/\*%Table\_Drawdowns(prices, asset=IBM, TOP=10, method=LOG, digits=5, dateColumn=DATE, outData=MyTableDrawdowns, printTable=PRINT);\*/

**[Table\_DrawdownsRatio](#tableDrawdownsRatio_TOC)** a table of drawdown-related ratios

**Description**

The table contains Calmar Ratio, Sterling Ratio, Burke Ratio, Pain Index, Pain Ratio, Ulcer Index, Martin Ratio.

**Usage**

%Table\_DrawdownsRatio(returns);

%Table\_DrawdownsRatio(returns, Rf=, scale=, method=, digits=, dateColumn=, outData=, printTable=);

**Arguments**

returns - Required. Data Set containing returns of the portfolio.

Rf - Optional. The value or variable representing the risk free rate of return. [Default=0]

scale - Optional. Number of periods in a year. {daily=252, monthly=12, quarterly=4, yearly=1}. [Default=1]

method - Optional. Specifies either DISCRETE or LOG chaining method {DISCRETE, LOG}. [Default=DISCRETE]

digits - Optional. Specifies number of digits displayed in the output. [Default=4]

dateColumn - Optional. Date column in Data Set. [Default=DATE]

outData - Optional. Specifies name of output Data Set of correlations. [Default="table\_DrawdownsRatio"]

printTable - Optional. Option to print output table. {PRINT, NOPRINT}. [Default= NOPRINT]

**Author**

Qiyuan Yang

**See Also**

[Burke\_Ratio](#name_BurkeRatio), [Calmar\_Ratio](#name_CalmarRatio), [Sterling\_Ratio](#name_SterlingRatio), [Pain\_Index](#name_PainIndex), [Pain\_Ratio](#name_PainRatio), [Ulcer\_Index](#name_UlcerIndex), [Martin\_Ratio](#name_MartinRatio)

**Example**

%let dir=C:\SVN\SAS\_Perf\_Anly;

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***Table\_DrawdownsRatio***(prices, Rf=, scale=, method=, digits=, dateColumn=, outData=, printTable=);

/\*Or\*/

/\*%Table\_DrawdownsRatio(prices, Rf=0, scale=252, method=LOG, digits=4, dateColumn=DATE, outData=MyTable, printTable=PRINT);\*/

[Table\_HigherMoments](#tableHigherMoments_TOC) Higher Moments summary: Statistics and stylized facts

**Description**

Summary of the higher moments and co-moments of the return distribution. Used to determine diversification potential. Also called “systematic” moments.

**Usage**

%table\_HigherMoments(Returns);

%table\_HigherMoments(Returns, dateColumn=, outData=, printTable=);

**Arguments**

Returns- required. Data Set containing returns.

dateColumn- specifies the date column in the data set. [Default= Date]

outData- output Data Set with “systematic” moments. [Default= “Higher\_moments”]

printTable- option to print table. {PRINT, NOPRINT}, Default= [NOPRINT].

**Author**

Dominic Pazzula, Carter Johnston

**See Also**

[BetaCoMoments](#name_BetaCo), [CoMoments](#name_CoMoments)

**Examples**

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***table\_HigherMoments(***prices);

/\*Or\*/

/\*%table\_HigherMoments(prices, dateColumn= DATE, outData=Higher\_Moments, printTable= PRINT);

**[Table\_InformationRatio](#tableInformationRatio_TOC)** Information Ratio summary: Statistics and stylized facts

**Description**

Table of the Tracking error, annualized tracking error, and information ratio.

**Usage**

%table\_InformationRatio(Returns, BM=);

%table\_InformationRatio(Returns, BM=, scale=, dateColumn=, outData=, printTable=);

**Arguments**

Returns- required. Data Set containing returns.

BM- Required. Specifies the variable name of benchmark asset or index in the returns data set.

scale- Optional. Number of periods in a year {any positive integer, ie daily scale=252, monthly scale=12, quarterly scale=4}. [Default=1]

dateColumn- specifies the date column in the data set. [Default= Date]

VARDEF- Optional. Specify the variance divisor, DF, degree of freedom, n-1; N, number of observations, n. {N, DF} Default= DF.

outData- output Data Set with information ratio and tracking error. [Default= “table\_InformationRatio”]

printTable- option to print table. {PRINT, NOPRINT}, [Default= NOPRINT].

**Author**

Dominic Pazzula, Carter Johnston

**See Also**

[Information\_Ratio](#name_InformationRatio), [TrackingError](#name_trackingError)

**Examples**

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***table\_InformationRatio(***prices, BM= SPY);

/\*Or\*/

/\*%table\_InformationRatio(prices, BM= DOW, scale= 252, dateColumn= DATE, outData=table\_InformationRatio, printTable= PRINT);

**[Table\_SpecificRisk](#tableSpecificRisk_TOC)** Specific Risk summary: Statistics and stylized facts

**Description**

Table of specific risk, systematic risk, and total risk of an asset or financial instrument. Total risk is assumed to be the standard deviation in this table.

**Usage**

%table\_SpecificRisk(Returns, BM=, Rf=);

%table\_SpecificRisk(Returns, BM=, Rf=, scale=, dateColumn=, outData=, printTable=);

**Arguments**

Returns- required. Data Set containing returns.

BM- required. Names variable containing returns of benchmark asset or index.

Rf- Either a constant or variable representing the risk free rate of return.

scale- Optional. Number of periods in a year {any positive integer, ie daily scale=252, monthly scale=12, quarterly scale=4}. [Default=1]

VARDEF- Optional. Specify the variance divisor, DF, degree of freedom, n-1; N, number of observations, n. {N, DF} Default= DF.

dateColumn- specifies the date column in the data set. [Default= Date]

outData- output Data Set with information ratio and tracking error. [Default= “table\_SpecificRisk”]

printTable- option to print table. {PRINT, NOPRINT}, Default= [NOPRINT].

**Author**

Dominic Pazzula, Carter Johnston

**See Also**

[Specific\_Risk](#name_SpecificRisk), [Systematic\_Risk](#name_SystematicRisk), [Standard\_Deviation](#name_StandardDeviation)

**Examples**

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***table\_SpecificRisk(***prices, BM= SPY);

/\*Or\*/

/\*%table\_SpecificRisk(prices, BM= DOW, Rf= 0.01/252, scale= 252, dateColumn= DATE, outData=table\_SpecificRisk, printTable= PRINT);

**[Table\_Stats](#tableStats_TOC)** Returns summary: Statistics and stylized facts

**Description**

Returns a table of basic statistics on a data set that match the period of the data passed in (e.g. monthly returns will return monthly statistics and daily returns will return daily stats…)

**Usage**

%table\_Stats(Returns);

%table\_Stats(Returns, alpha=, outData=, dateColumn=, digits=, printTable=);

**Arguments**

Returns- required. Data Set containing returns.

Alpha- specifies the significance level for the mean. [Default= 0.05]

dateColumn- specifies the date column in the data set. [Default= Date]

outData- output Data Set with related statistics. [Default= “table\_SpecificRisk”]

digits= specifies the number of digits to display in the output data set.

VARDEF- Optional. Specify the variance divisor, DF, degree of freedom, n-1; N, number of observations, n. {N, DF} Default= DF.

printTable- option to print table. {PRINT, NOPRINT}, Default= [NOPRINT].

**Author**

Dominic Pazzula, Carter Johnston

**Examples**

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***table\_Stats(***prices);

/\*Or\*/

/\*%table\_Stats(prices, alpha= 0.05, outData= Stats, dateColumn= DATE, digits= 4,printTable= PRINT);

**[Table\_Variability](#tableVariability_TOC)** Variability summary: Statistics and stylized facts

**Description**

Table of mean absolute deviation, monthly standard deviation, and annualized standard deviation.

**Usage**

%table\_Variability(Returns, scale=);

%table\_Variability(Returns, scale=, dateColumn=, outData=, printTable=);

**Arguments**

Returns- required. Data Set containing returns.

scale- required. Denotes the scale for which to annualize the standard deviation.

VARDEF- Optional. Specify the variance divisor, DF, degree of freedom, n-1; N, number of observations, n. {N, DF} Default= DF.

dateColumn- specifies the format of the date column in the data set. [Default= Date]

outData- output Data Set with related statistics. [Default= “table\_Variability”]

printTable- option to print table. {PRINT, NOPRINT}, Default= [NOPRINT].

**Author**

Dominic Pazzula, Carter Johnston

**Examples**

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***table\_Variability(***prices, scale= 252);

/\*Or\*/

/\*%table\_Variability(prices, scale= 252,dateColumn= DATE, outData= variability\_table, printTable= PRINT);

**[TrackingError](#trackingError_TOC)** Calculate tracking error of returns against a benchmark

**Description**

The tracking error is the measure of the unexplained portion of performance relative to a benchmark asset or index.

**Details**

The tracking error is calculated by taking the square root of the average of the squared deviations between the investment’s returns and the benchmark’s returns, then multiplying the result by the square root of the scale of returns.

where

**Usage**

%TrackingError(Returns, BM=, scale=);

%TrackingError(Returns, BM=, annualized=, scale=, dateColumn=, outData=);

**Arguments**

Returns - required. Names the data set containing time series data of asset returns.

BM- required. Names the benchmark asset or index variable in the data set.

Annualized- option to annualize the tracking error. [Default= FALSE]

Scale- optional. The number of periods in a year (daily scale= 252, monthly= 12) if annualized= TRUE. [Default= 1].

VARDEF- Optional. Specify the variance divisor, DF, degree of freedom, n-1; N, number of observations, n. {N, DF} Default= DF.

dateColumn- specifies the date column in the data set. [Default= Date]

outData- output data set with tracking errors. [Default= tracking\_error]

**Author**

Dominic Pazzula, Carter Johnston

**See Also**

[Information\_Ratio](#name_InformationRatio)

**Examples**

%let dir=C:\SVN\SAS\_Perf\_Anly;

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***TrackingError***(prices, BM= SPY);

/\*Or\*/

/\*%TrackingError(prices, BM= SPY, scale= 252, annualized= TRUE, dateColumn= Date, outData= tracking\_error);\*/

**[Treynor\_Ratio](#TreynorRatio_TOC)** calculate Treynor Ratio of return over beta or systematic risk

**Description**

Calculates the regular Treynor Ratio, which is the asset excess return divided by asset beta. Asset beta comes from CAPM model and a market benchmark needs to be specified. Option to calculate modified Treynor Ratio is available and the denominator will be replaced by systematic risk.

**Details**

Treynor Ratio is very similar to Sharpe Ratio in a way that both ratios measure portfolio performance with respect to return and risk. Treynor Ratio uses the relative market risk and is thus more suitable for well-diversified portfolio. Sharpe Ratio uses the standard deviation and can be applied to all portfolios.

The systematic risk is defined by Bacon (2008) as the product of beta by market risk.

**Usage**

%Treynor\_Ratio(returns=, BM=);

%Treynor\_Ratio(returns=, BM=, scale=, method=, modified=, dateColumn=, outData=);

**Arguments**

returns - Required. Data Set containing returns with option to include risk free rate variable.

BM - Required. Specifies the variable name of benchmark asset or index in the returns data set.

Rf - Optional. The value or variable representing the risk free rate of return. [Default=0]

scale - Optional. Number of periods in a year {any positive integer, ie daily scale= 252, monthly scale= 12, quarterly scale= 4}.

method - Optional. Specifies either DISCRETE or LOG chaining method {DISCRETE, LOG}. [Default=DISCRETE]

VARDEF - Optional. Specify the variance divisor, DF, degree of freedom, n-1; N, number of observations, n. {N, DF} [Default= DF]

modified - Optional. Specifies either regular or modified Treynor Ratio {FALSE, TRUE}. [Default=FALSE]

dateColumn - Optional. Date column in Data Set. [Default=DATE]

outData - Optional. Output Data Set with Treynor ratios. [Default="TreynorRatio"]

**Author**

Qiyuan Yang

**See Also**

Sharpe Ratio, Systematic Risk, CAPM\_alpha\_beta.

**Example**

%let dir=C:\SVN\SAS\_Perf\_Anly;

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***Treynor\_Ratio***(prices, BM=IBM);

/\*Or\*/

/\*%Treynor\_Ratio(prices, BM=IBM, Rf=0, scale=1, method=LOG, modified=FALSE, dateColumn=DATE, outData=MyTreynorRatio);\*/

**[Ulcer\_Index](#UlcerIndex_TOC)** calculate Pain Index

**Description**

Calculate Ulcer Index devised by Perter Martin in 1987. Different from standard deviation that takes into account both up and down movements, Ulcer Index takes downside drawdowns.

**Details**

The Pain Index can be calculated as:

Where n denotes the number of observations from the data set, and Dt is the tth drawdown.

**Usage**

%Ulcer\_Index(returns);

%Ulcer\_Index(returns, method=, dateColumn=, outData=);

**Arguments**

returns - Required. Data Set containing returns.

method - Optional. Specifies either DISCRETE or LOG chaining method {DISCRETE, LOG}. [Default=DISCRETE]

dateColumn - Optional. Date column in Data Set. [Default=DATE]

outData - Optional. Output Data Set with ulcer index. [Default="UlcerIndex"]

**Author**

Qiyuan Yang

**Example**

%let dir=C:\SVN\SAS\_Perf\_Anly;

libname input "&dir";

/\*Include SASPerformanceAnalytics\*/

%include "&dir\macro\\*.sas" /nosource;

**data** prices;

set input.prices;

**run**;

%***return\_calculate***(prices);

%***Ulcer\_Index***(prices);

/\*Or\*/

/\*%Ulcer\_Index(prices, method=LOG, dateColumn=DATE, outData=MyUlcerIndex);\*/