OCTARISK Documentation

version 0.2.0

Stefan Schloegl (schinzilord@octarisk.com)

This is the documentation for the market risk measurement project OCTARISK.

Table of Contents

OCTARISK Documentation

This manual is for the market risk measurement project OCTARISK (version 0.2.0).

1 Introduction

1.1 Why quantifying market risk?

This ongoing project is made for all investors who want to dig deeper into their portfolio than just looking at the yearly profit or loss. Although most financial reports of more sophisticated brokers contain risk measures like standard deviations, the volatility alone cannot cover the risk inherent in non-linear financial products like options. Moreover, potential investors care about their portfolio values under certain market conditions, e.g. they want to compare their perceived personal stress levels during the financial crisis but with the financial instruments in their portfolio losses during stress scenarios.

If questions like

- What happens to my portfolio value, if the ECB increases the interest rate by 100bp?
- What happens to my portfolio value, if an emerging country is defaulting and the US Dollar increases in value?
- How would my portfolio perform, if we have a new crash comparable to Black Friday 1987?

are of potential interest for you, then the OCTARISK market risk project might be satisfying your needs for a professional risk modelling framework.

Since most investors do not give excessive credits to debtors or bear operational or liquidity risks, the OCTARISK project focuses on market risk only - all remaining types of risk which are relevant for your investment portfolio: equity risk, interest rate risk, volatility risk, commodity risk, ...

For the assessment of these market risk types, a sophisticated full valuation approach with Monte-Carlo based value-at-risk and expected shortfall calculation is performed. The underlying principles are state-of-the-art in financial institutions and are used in internal models to fulfill the requirements set by regulators (for Basel III and Solvency 2). The important concepts are adopted, the unnecessary overhead was skipped - resulting in a fast, lightweight yet flexible approach for quantifying market risk.

1.2 Features

The OCTARISK quantifying market risk projects features

- a full valuation approach for financial instruments
- Monte-Carlo method for scenario generation of underlying risk factors
- simple input interface via static and dynamic files
- a processable report incl. graphical representation of portfolio profits and losses as well as an overview over the riskiest instruments and positions (see See Section 3.4 [Output files], page 23 for examples)
- an easily customizable framework based on full implementation on Octave with compact source code

1.3 Prerequisites

The only requirement is GNU Octave (tested for versions > 4.0) with installed financial and statistical package and hardware with minimum of 5Gb of memory. Calculation time decreases significantly while using optimized linear algebra packages of OpenBLAS and LAPACK (or comparable). For automatic processing of the input data (e.g. to get actual market data from Quandl or Yahoo finance), to make the parameter estimation as well as process the report files, some programming language like Perl, Python and a running LaTeX environment are recommended, but not required.

Nevertheless, a basic understanding of a high level programming language like Octave is required to adjust the source code and to customize the calculation. Furthermore, a thorough understanding of financial markets, instruments and valuation will be needed in order to select appropriate models and to interpret results. The implemented models and the underlying concepts are explained in detail for example in following literature:

Risk Management and Financial Institutions, John C. Hull, 2015
Paul Wilmott on Quantitative Finance, 2nd Edition, Paul Wilmott, 2006
Options, Futures and other Derivatives, 7th Edition, John C. Hull, 2008

The next chapter contains the background and details needed for running the market risk valuation and aggregation software and understanding the risk measures.

2 User guide

This chapter gives a general introduction to market risk and how the market risk is captured by OCTARISK. Thereby the following convention is made: market movement means the movement around the mean value or, in other words, the possible deviation from the expected value as time passes. Stronger movement around the expected value means more risk and results in a higher standard deviation, the most important statistic parameter for capturing market risk.

2.1 Introducing market risk

Typically, market risk can be divided into several sub types. Most of the splitting is obvious, but some of the more exotic risk types remain very broad. The ideal breakdown depends heavily on the specific portfolio whose market risk shall be quantified. A basic approach fitting most portfolios of private investors, not too broad, not too granular, is chosen in OCTARISK. The following types of risk are defined:

- *FX*: Forex risk captures the market movements of exchange rate values. These movements relative to the reporting currency (in our case EUR) affect financial assets in foreign currencies only.
- *EQ*: Equity risk is related to the movement of equity markets. It will be typically broken down into sub-categories like countries, regions or other types of aggregation (e.g. developed market, emerging market, frontier markets).
- *COM*: Commodity risk. This is a rather broad type of risk, often correlated to other risk types (e.g. equity). Commodity risk tries to capture the movement of spot and future / forward commodity prices, as well as commodity linked equities.
- *IR*: Interest rate risk is related to the movement of interest rate values. This is the most important type of risk for cash flow bearing instruments.
- *RE*: Real estate risk, where typically it will be distinguished between movements of market values of REITs (Real Estate Investment Trusts) and housing prices. This risk type can also be broken down into sub-categories like different countries, regions or other categories (e.g. developed markets, emerging markets).
- *VOLA*: Implied volatility risk, e.g. the anticipated future movement of implied volatility of equity instruments or swaps.
- *ALT*: Alternative market risk, used as a container for every other type of risk not already captured (e.g. Bitcoins, infrastructure)

The OCTARISK projects focuses on these seven risk types. For each risk type, appropriate risk factors can be chosen. One risk factor is a typical representative of a particular risk type. Most often, several risk factors are needed to describe the granular behavior of a market risk type, e.g. it is necessary to describe the specific characteristics of countries, regions or currencies. For example, two risk factors can be used to describe the international equity market movements: Developed markets and emerging market. If desired, developed market can be further split into North America, Europe and Asia to describe risk diversification effects between these broad regions. After the selection of risk factors, stochastic models are chosen, calibrated and subsequently used for scenario generation.

2.2 Market risk measures

2.2.1 Value-at-Risk

Value-at-risk (VAR) is defined as the monetary loss which the portfolio won't exceed for a specific probability on a certain time horizon. As an example, a 250 trading day (one calender year) VAR of 1000 EUR at the 99% confidence interval means there is a probability of 99% that the portfolio loss within one year (250 trading days) is equal to or less than 1000 EUR. It is important to note that no forecast is made for the possible loss which can occur in 1% of the remaining cases. Furthermore, within a proper calibrated risk setup one **must** expect a loss greater than the VAR amount in 1% of all cases, that means a loss of more than 1000 EUR will occur in two to three trading days per year. Otherwise, if there are no trading days observed where the loss is greater than predicted by VAR, the risk is overstated, leaving room for better usage of risk capital.

2.2.2 Expected shortfall (ES)

Expected shortfall is an additional risk measure which is defined as the arithmetic average (mean) loss in the remaining tail of the sorted profit and loss distribution of all simulated MC scenarios, which are not covered by the 99% VAR. The ES should always be seen in context of the VAR and is a more coherent risk measure, which can make stronger predictions about diversification benefits of portfolios.

2.3 Scenario generation

A scenario is a specific set of shocks to risk factors. Typically, the directions of the shocks are correlated, and the value of the shock is dependent on the stochastic properties of the risk factors (e.g. volatility or mean reversion parameters). Although these properties can be also chosen customary, for evaluating risk measures like value-at-risk these parameters are typically extracted from past, real market movements.

2.3.1 Stochastic models

In order to describe movements of risk factors in time, a connection has to be made between statistical behavior of real time series and stochastic processes for modeling synthetic time series. OCTARISK concentrates on three stochastic processes: Wiener, Ornstein-Uhlenbeck and root-diffusion processes.

2.3.1.1 Random walk and the Wiener process

For the Wiener process, two different possible definitions exist. In the first case, both the drift and the normally-distributed random number W_t (the so called Wiener process) are proportional to the variable at the former time step, resulting in the process S_t which satisfies the following stochastic differential equation:

$dS_t = mu * S_{t-1} * dt + sigma * S_t * dW_t$
The following analytic solution to this stochastic differential equation is derived:

$S_t = S_0 * exp((mu - sigma^2/2) * t + sigma * W_t)$
This solution ensures positive values at all time steps.

In the second case, a continuous time random walk with independent, normally-distributed random numbers independent of the variable at the former time step is given by the following stochastic differential equation:
$dS_t = mu * dt + sigma * W_t$
This process shows self-similarity and scaling behavior.

2.3.1.2 Ornstein-Uhlenbeck process

The Wiener process can be extended to incorporate a serial dependency (like a memory) - tomorrows values are dependent on the level of todays values. The Ornstein-Uhlenbeck (OU) process has a mean-reversion term, which is directly proportional to the difference of the actual value from the mean reversion level:
$dX_t = mr_{rate} * (mr_{level} - X_{t-1}) * dt + sigma * dW_t$
where the mean reversion rate mr_{rate} can be seen as a proportional parameter of a restoring force. The increments dX_t tend to point to the mean-reversion level mr_{level}. The result of the formula is an addditive term to the risk factor depending on the past level and a stochastic term (modeled by the Wiener process).

2.3.1.3 Square-root diffusion process

In order to exclude negative values in the OU mean-reversion process, an additional repelling force is needed, which ensures that the level of the stochastic variables stays away from zero. The square-root diffusion process (SRD) has an additional term, which is multiplied with the standard deviation and the random variable. This term is identified as the square root of the variable at the former time step:
$dX_t = mr_{rate} * (mr_{level} - X_{t-1}) * dt + sigma * sqrt(X_{t-1}) * dW_t$
Negative values for the variables are excluded if the following equation is fulfilled:
$2 * mr_{rate} * mr_{level} >= sigma$

2.3.2 Financial models

The stochastic models which have been presented in the last section, are used as basis for financial models. In order to map stochastic processes to financial models, properties of financial models are identified and subsequently stochastic models chosen in order to generate simulated time series of risk factors with appropriate and desired behavior.

2.3.2.1 Geometric Brownian motion

The standard financial model for equity, real-estate and commodity risk factors is a geometric Brownian motion (GBM), which utilizes the extended Wiener process, where the drift and random variable are proportional to the risk factor value. Due to this proportionality the time series can not reach zero and the modeled risk factors values always stay positive. This is reflected in the real world behavior of equity or commodity prices, where the intrinsic value of these assets cannot fall below zero.

2.3.2.2 Brownian motion

In a Brownian motion (BM) model, the time-series increments are a function of drift, time and standard deviation, but not dependent on the actual level of the financial variable. For certain types of risk factors (e.g. interest rates), one assumes a Brownian motion so that the modeled price movements are given as additive shocks which are completely independent on the actual risk factor value. Therefore, negative values of the modeled risk factors are allowed.

2.3.2.3 Vasicek model

In the long run, the market movements of exchange rates and interest rates seem to be mean-reverting. This behavior can be modeled by a Ornstein-Uhlenbeck process resulting in a so called one factor short rate model proposed by Vasicek. The Ornstein-Uhlenbeck process allows for negative values, which reflects the real world behavior of short rates since the financial crisis, where interest rates of AAA-rated government bonds had negative yields for at east some time.

2.3.2.4 Cox-Ingersoll-Ross and Heston model

If one doesn't want to allow negative values for interest rates or other mean-reverting risk factors, a square-root diffusion process can be chosen as stochastic model. This results in the short-rate model of Cox-Ingersoll-Ross or the Heaston model for modelling at-the-money volatility used in option pricing.

2.3.3 Parameter estimation

Once an appropriate model is chosen for the risk factor, one has to define input parameters for the stochastic differential equations. One approach is to use historical data to estimate statistic parameters like volatility, correlations or mean-reversion parameters. Another approach is to apply expert judgment in selecting input parameters for the models. OCTARISK uses the specified parameters to generate Monte-Carlo scenarios - the selection of the parameter estimation approach is up to the reader.

Typically, parameters are extracted from historical time series on weekly or monthly data on the past three to five years. Unfortunately, availability of historical time series for all risk factors is one of the main constraints in parameter estimation for private investors. Most often, one has to overcome problems of missing data and the need for interpolation or extrapolation. In future versions scripts for parameter estimation will be provided.
In an ideal world, at first appropriate risk types and risk factors are selected, then the validation of a stochastic model is performed and the appropriateness of the model and the assumptions (like length of historical time series) is verified in back-tests. Nevertheless, no stochastic model can completely describe the financial markets, giving rise to model error (both in parameter estimation and model selection).

2.3.4 Monte-Carlo Simulation

A Monte-Carlo approach is chosen to generate the risk factor shocks in all scenarios. Therefore a risk factor correlation matrix (e.g. estimated from historical time series) and additional parameters describing statistical distributions can be used to generate random numbers, which are utilized as input parameters to stochastic models.

In order to account for non-normal distributions and higher order correlation effects in the Monte-Carlo simulation, a copula approach is chosen to generate dependent, correlated random numbers. In a first step, the input correlation matrix is used to generate normally distributed, correlated random numbers with zero drift and unit variance. In a next step, either a Gaussian copula or a student-t copula is utilized to transform the normally distributed random variables to uniformly distributed random variables, while either the linear correlation dependence (for Gaussian copulas) or additionally the non-linear dependence structures (for student-t copulas) is preserved. In a last step, these random numbers are incorporated into a function which chooses for each risk factor the appropriate distribution in a certain way, that standard deviation, skewness and kurtosis are matched with the input parameters. Therefore the Pearson distribution system is used as a basis for generation of random variables. Subsequently, in each of the Monte-Carlo scenarios a specific set of correlated risk factor shocks, dependent on the selected financial models, is generated, while the marginal distributions have desired standard deviation, skewness and kurtosis.

A further potential problem is the model error from the Monte-Carlo method. Only a limited number of Monte-Carlo scenarios can be generated and valuated (in the range of 10000 to 100000), thus leaving space for not represented scenarios which could alter the risk measures.

2.3.5 Stress testing

A complementary method to the stochastic scenario generation is to directly define shocks to risk factors. This scenario analysis is normally done in order to calculate the portfolio behavior in well known historic scenarios (like Financial Crisis 2008, devaluation of Asian currencies in the mid 90s, terrorist attack on 9/11, Black Friday in October 1987) or in scenarios, where one only is interested in the behavior of the portfolio value in isolated shocks (like all equities decline by 30pct in value, a +100bp parallel shift in interest rates). Possible sources of scenarios are provided by regulators and can be easily adapted for personalized stress scenarios.

2.4 Instrument valuation

After the scenarios are generated, one has to calculate the behavior of financial instruments to movements in the underlying risk factors. Therefore two different approaches are chosen: the sensitivity approach applies relative valuation adjustments to the instrument base values proportional to the defined sensitivity. The valuation adjustments are derived from movements in value of the underlying risk factors. Secondly, in a full valuation approach, the scenario dependent input parameters are fed into a pricing function which (re)calculates the new absolute value of the instrument in each particular scenario.

2.4.1 Sensitivity approach

The sensitivity approach is performed for all instruments which have a linear dependency on the movement of underlying risk factor values, or where not enough data or no appropriate pricing function exists for a full valuation approach.

2.4.1.1 Linear dependency on risk factor shocks

For the risk assessment of equities, commodities or real estate instruments it is appropriate to take only the linear dependency on risk factor movements into account. Each risk factor has sensitivities to underlying risk factors. An instrument inherits the amount of shock proportional to the defined sensitivity value from the underlying risk factors.

A typical example is a developed market exchanged traded fund (ETF) with exposure to North America, Europe and Asia. The ETF will follow the price movements of these three risk factors, so the sensitivities are simply the relative exposure to the three equity markets (e.g. 0.5, 0.3 and 0.2). These sensitivities are then used to calculate the weighted shock that is applied to the actual instrument value. The same principle holds for single stock, where sensitivities to appropriate risk factors and to an idiosyncratic risk term (an uncorrelated random number) can be selected. A useful method for calibration is the multi-dimensional linear regression. The resulting betas from this regression can be taken for the sensitivities to regressed risk factors. The remaining alpha and estimation error resembles the sensitivity to the idiosyncratic risk. The exposure to the uncorrelated random number can be derived from one minus the adjusted R_square of the regression. Since the R_square gives the amount of variability that is explained by the regression model, one minus R_square is equal to the amount of uncorrelated random fluctuations which are not covered by the input parameters.

2.4.1.2 Approximation with sensitivity approach

For instruments with insufficient information one can also choose the sensitivity approach. One example are funds consisting mainly of bonds. Without look-through, one has no information about the exact cash flows of the underlying bonds. Instead, often the duration (and convexity) of the fund is known. These two types of sensitivity (duration and convexity) can then be used to calculate the change in value to interest rate shocks. Therefore, the absolute interest rate shock at the node, which equals the Macaulay duration, is calculated as absolute difference compared to the base rate. This change in interest rate level (dIR) is directly transformed to a relative value shock (dV) by the formula
$dV = duration * dIR + convexity * dIR^2$
incorporating both sensitivities.

2.4.2 Full valuation approach

The core competency of a quantitative risk measurement project is full valuation, where the absolute value of financial instruments is calculated from raw input parameters by a special pricing function. Some input parameters to the pricing function are scenario dependent, other are inherent to the instrument. The most important input parameters have to be modeled by stochastic processes and can subsequently be fed into the pricing function, where the new, scenario dependent absolute value of the instrument is calculated. At the moment, the following full valuation pricing functions are implemented in OCTARISK:

2.4.2.1 Option pricing

European plain-vanilla options are priced by the Black-Scholes model (See Section 4.26 [option_bs], page 36). The Black-Scholes equation provides a best estimate of the option price. The underlying financial instrument and implied volatility are modeled as risk factors in order to calculate the new option price in each scenario. The risk free rate will be also made scenario dependent in order to capture the interest rate sensitivity of the option price. Further information is provided by numerous textbooks.

For American options a more sophisticated model has to be used for pricing. Unfortunately, binomial models (like the Cox-Rubinstein-Ross model) or finite-difference models are not feasible for a full valuation Monte-Carlo based approach, since the computation time for a large amount of time steps and MC scenarios is too high due to missing parallelization opportunities. Instead, a Willow-Tree model is implemented to price American options. Within that model, instead of using the full binomial tree with increasing number of nodes per time step, a constant number of nodes at each pricing time step is utilized to approximate the movements of the underlying price. With optimized transition probabilities, the whole model relies on a smaller amount of total nodes which significantly decreases computation time and lowers memory consumption (See Section 4.27 [option_willowtree], page 37 implementation for further details).

A calibration is performed to align the model price based on provided input parameters with the observed market price. This calibration calculates an implied spread which is added to the modeled volatility as a constant offset.

The implied volatility itself is dependent on the option strike level and time to maturity (term). In order to grasp that behavior, the so called volatility smile is modeled by a moneyness vs. term volatility surface, where changes in the spot price lead to moneyness changes. Therefore, the actual implied volatility behavior (at-the-money implied volatility vs. moneyness vs. term) of the market is preserved for the pricing.

2.4.2.2 Swaption pricing

European plain-vanilla swaptions are priced via the closed form solution of the Black-76 model or the Bachelier model. (See Section 4.34 [swaption_black76], page 42 and See Section 4.33 [swaption_bachelier], page 41 for details). Again, a calibration is performed to align the model price based on provided input parameters with the observed market price. The calibrated implied volatility spread is subsequently added to the modeled volatility as a constant offset. The volatility smile for the specific term is also given by volatility cubes. The interest rate implied volatility can be set up by three axes: underlying tenor, swaption term and moneyness. For each of these combinations an implied volatility can be set. Again, the smile is preserved during MC and stress scenarios. There is only one risk factor (at-the-money implied volatility) modelled, which provides a scenario dependent constant offset for the volatility cube.

2.4.2.3 Forward pricing

Equity and Bond forwards can be valuated. Therefore market indizes can be set up, which serve as underlyings for the forwards. The value of the forward is calculated as the payoff at maturity (underlying value minus strike) discounted back to the valuation date (See Section 4.28 [pricing_forward_oop], page 38 for details).

2.4.2.4 Cash flow instrument pricing

Cash flow instruments are specified by the following sets of variables: cash flow dates and corresponding cash flow values. Moreover, each cash flow instrument has an actual market price and an underlying interest rate curve, which has to be provided as a separate risk factor. Before the full valuation can be carried out, the spread over yield is calculated to align the observed market price with the value given by the pricing function. The spread over yield is then assumed to be a constant offset to the scenario dependent interest rate spot curve. For each scenario, all cash flows are discounted with the appropriate interest rate and spread curve. The present value is then given by the sum of all discounted cash flows. Credit spreads are modeled as separate risk factors, thus capturing credit spread risk for cash flow instruments.

2.4.2.5 Bond instrument pricing

The Bond instrument class covers the full spectrum of plain vanilla bond instruments: fixed rate bonds, floating rate notes, fixed and floating swap legs, fixed rate amortizing bonds. During pricing, at first the cash flows are rolled out. The forward rates for calculation of floating payments are scenario dependent. After the rollout is done, a spread over yield is calibrated in order to match the market price with the theoretical value. For calculation of the net present value of the bonds, a discount curve and a spread curve can be set. If the curves have attached risk factors, the pricing will be fully scenario dependent.

2.4.3 Synthetic instruments

In order to model a fixed share combination of instruments, the synthetic class was introduced. The value of the synthetic instrument is calculated as the linear combination of all underlying instrument. Synthetic instruments can be used to e.g. model portfolios with full look-through or to combine fixed and floating legs to swaps. Moreover, more complex instrument including option behaviour can be modeled, if e.g. a bond and a option is combined to an instrument with embedded call / put optionality.

2.5 Aggregation

After the valuation of all instruments in each scenario, one has to aggregate all parameters of instruments contained in a fund. Therefore all fund position values are derived by multiplying the position size with the instrument value in each scenario. The resulting sorted fund profit and loss distribution is then used to calculate the value-at-risk at fund level.
If less than 50001 MC scenarios are used in OCTARISK, the VAR is smoothened by a Harrell-Davis estimator (See Section 4.15 [harrell_davis_weight], page 32 for details). A weighted average of the scenarios around the confidence interval scenarios is calculated. The HD VAR shall reduce the Monte-Carlo error.

2.6 Reporting

After the aggregation of instrument data to fund level, a risk report is generated. The report contains VAR on position level, the riskiest instruments and positions per fund, as well as the diversified (with observed correlation) and undiversified (correlation set to 1) VAR and ES on fund level. Moreover, stress test results per fund, profit and loss distributions and histograms on both the one day and 250 day time horizon are generated (See Section 3.4 [Output files], page 23 for examples)

3 Developer guide

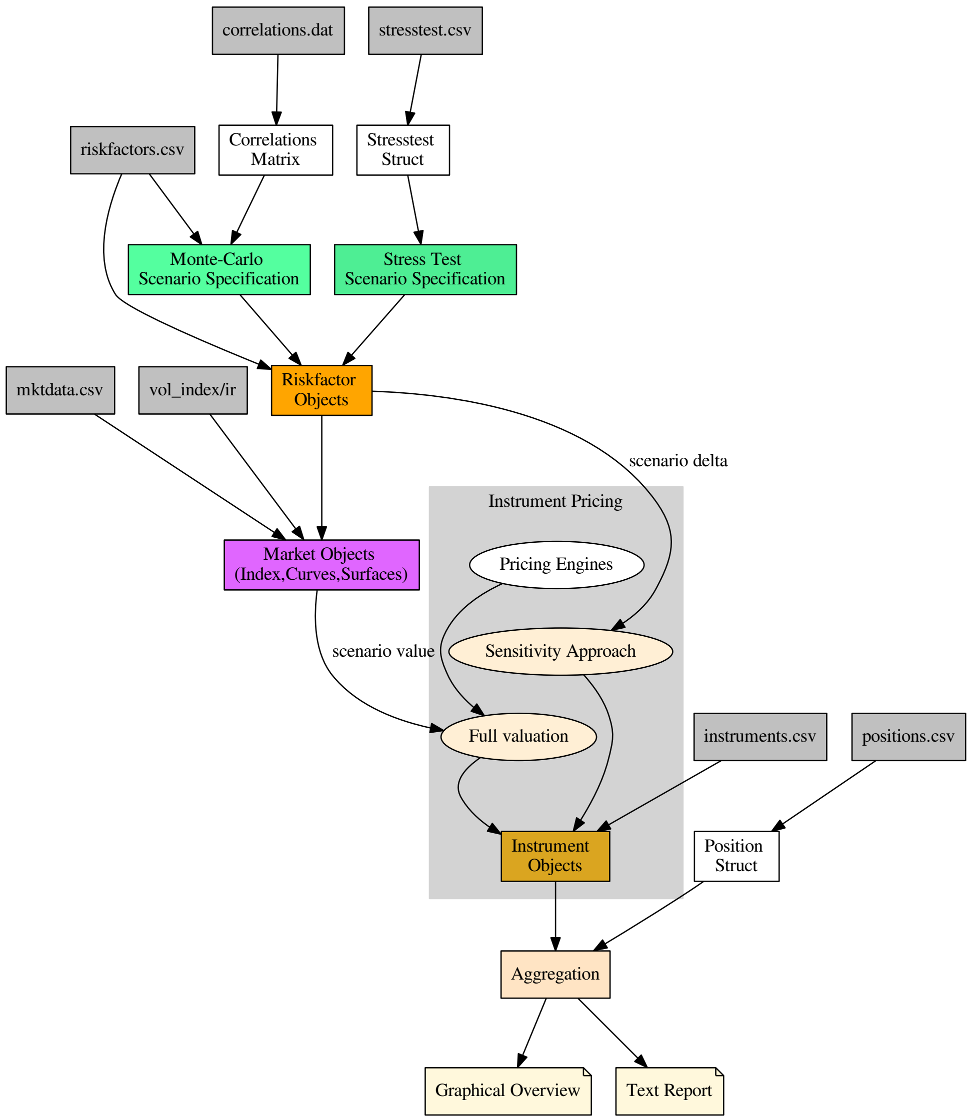
This chapter describes the actual implementation of the project. All calculation steps and the input and output files will be described in detail as well as examples are provided.

3.1 Implementation concept

A lightweight implementation concept was chosen for OCTARISK. One script is responsible for the complete work flow from input file parsing until aggregation and report printing. This script calls subfunctions to parse input data and construct objects, generate scenario dependent input values and call appropriate pricing functions.

3.1.1 Process overview

The following process summarizes the complete work flow:

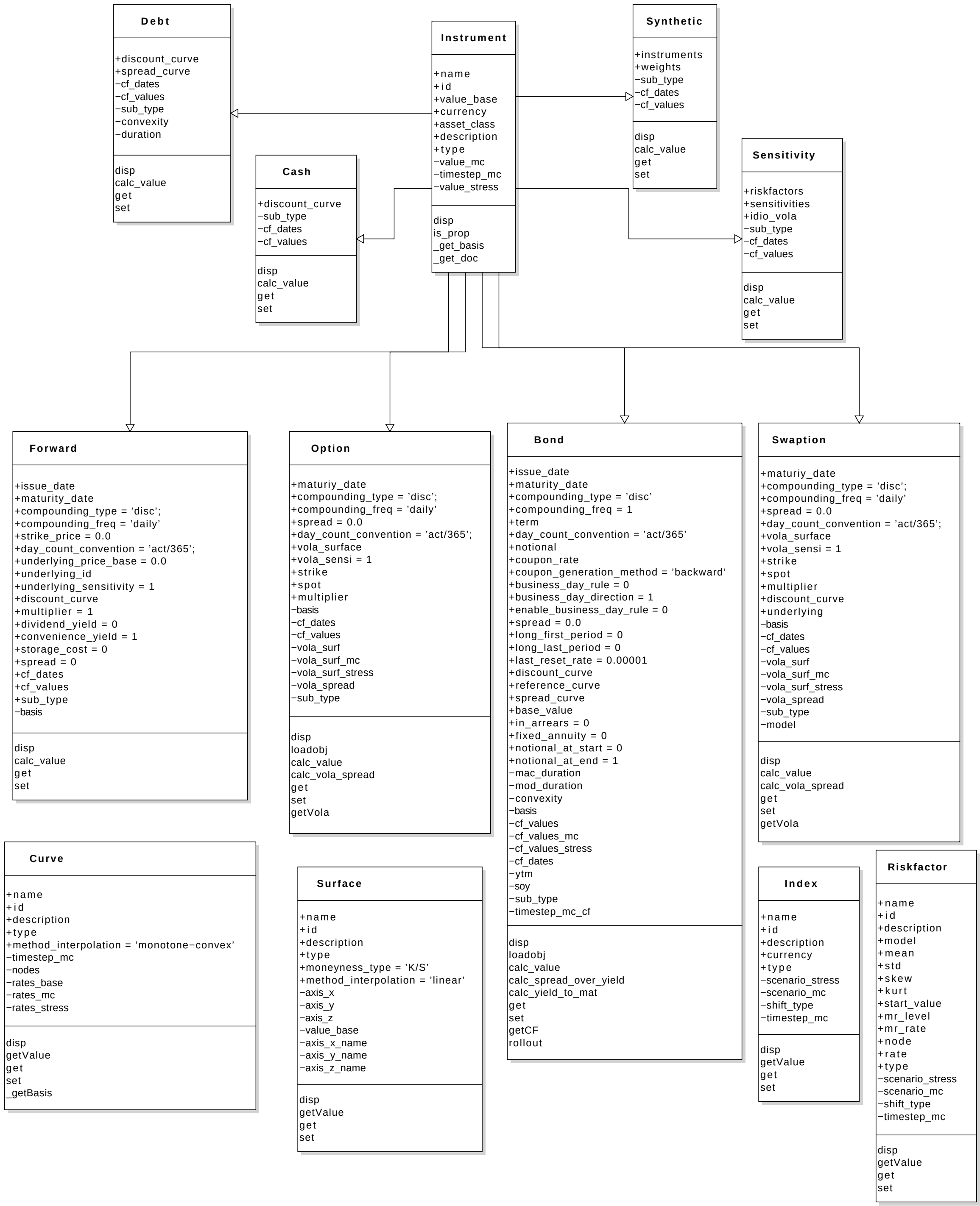


OCTARISK process

Input files (See Section 3.3 [Input files], page 17) are parsed into structures matrizes and objects.After parsing of these input files, Monte-Carlo and stress test scenarios are generated taking into account the correlation matrix and marginal risk factor distributions as well as custom stress test scenario configurations. The scenario dependent shocks (delta values) are then stored for each risk factor object. After the scenario generation the scenario shocks are applied to all market data objects (mainly indizes, curves and exchange rates) which have attached risk factors. Scenario dependent values for each market data object are calculated by taken into account the scenario dependent shock, the market data base value and the risk factor stochastic model. Now, the instrument pricing is ready to start. For each instrument object the product type dependent calculation rule is triggered. If the sensitivity approach is chosen, scenario shocks (delta values) are applied to the instrument base value. In case of a full valuation approach, the instrument is priced with appropriate pricing engines taking into account all required market objects. A mark-to-market procedure is applied, which results in equal theoretical and market base values (by setting an appropriate spread over yield or volatility spread). After all instruments have been priced, the aggregation starts for all portfolios and their positions. Final results are printed in graphical and text based reports reflecting the market risk measures.

3.1.2 Class diagram

OCTARISK was set up in an object oriented programming style for all objects like instruments, risk factors, curves, indizes and surfaces. Inside the methods of the aforementioned classes, pricing or interpolation functions are called. The following class diagram gives an overview of all classes:


Debt
+discount_curve
+spread_curve
−cf_dates
−cf_values
−sub_type
−convexity
−duration
disp
calc_value
get
set
Instrument
+name
+id
+value_base
+currency
+asset_class
+description
+type
−value_mc
−timestep_mc
−value_stress
disp
is_prop
_get_basis
_get_doc
Synthetic
+instruments
+weights
−sub_type
−cf_dates
−cf_values
disp
calc_value
get
set
Cash
+discount_curve
−sub_type
−cf_dates
−cf_values
disp
calc_value
get
set
Sensitivity
+riskfactors
+sensitivities
+idio_vola
−sub_type
−cf_dates
−cf_values
disp
calc_value
get
set
Forward
+issue_date
+maturity_date
+compounding_type = 'disc';
+compounding_freq = 'daily'
+strike_price = 0.0
+day_count_convention = 'act/365';
+underlying_price_base = 0.0
+underlying_id
+underlying_sensitivity = 1
+discount_curve
+multiplier = 1
+dividend_yield = 0
+convenience_yield = 1
+storage_cost = 0
+spread = 0
+cf_dates
+cf_values
+sub_type
−basis
disp
calc_value
get
set
Option
+maturiy_date
+compounding_type = 'disc';
+compounding_freq = 'daily'
+spread = 0.0
+day_count_convention = 'act/365';
+vola_surface
+vola_sensi = 1
+strike
+spot
+multiplier
−basis
−cf_dates
−cf_values
−vola_surf
−vola_surf_mc
−vola_surf_stress
−vola_spread
−sub_type
disp
loadobj
calc_value
calc_vola_spread
get
set
getVola
Bond
+issue_date
+maturity_date
+compounding_type = 'disc'
+compounding_freq = 1
+term
+day_count_convention = 'act/365'
+notional
+coupon_rate
+coupon_generation_method = 'backward'
+business_day_rule = 0
+business_day_direction = 1
+enable_business_day_rule = 0
+spread = 0.0
+long_first_period = 0
+long_last_period = 0
+last_reset_rate = 0.00001
+discount_curve
+reference_curve
+spread_curve
+base_value
+in_arrears = 0
+fixed_annuity = 0
+notional_at_start = 0
+notional_at_end = 1
−mac_duration
−mod_duration
−convexity
−basis
−cf_values
−cf_values_mc
−cf_values_stress
−cf_dates
−ytm
−soy
−sub_type
−timestep_mc_cf
disp
loadobj
calc_value
calc_spread_over_yield
calc_yield_to_mat
get
set
getCF
rollout
Swaption
+maturiy_date
+compounding_type = 'disc';
+compounding_freq = 'daily'
+spread = 0.0
+day_count_convention = 'act/365';
+vola_surface
+vola_sensi = 1
+strike
+spot
+multiplier
+discount_curve
+underlying
−basis
−cf_dates
−cf_values
−vola_surf
−vola_surf_mc
−vola_surf_stress
−vola_spread
−sub_type
−model
disp
calc_value
calc_vola_spread
get
set
getVola
Curve
+name
+id
+description
+type
+method_interpolation = 'monotone−convex'
−timestep_mc
−nodes
−rates_base
−rates_mc
−rates_stress
disp
getValue
get
set
_getBasis
Surface
+name
+id
+description
+type
+moneyness_type = 'K/S'
+method_interpolation = 'linear'
−axis_x
−axis_y
−axis_z
−value_base
−axis_x_name
−axis_y_name
−axis_z_name
disp
getValue
get
set
Index
+name
+id
+description
+currency
+type
−scenario_stress
−scenario_mc
−shift_type
−timestep_mc
disp
getValue
get
set
Riskfactor
+name
+id
+description
+model
+mean
+std
+skew
+kurt
+start_value
+mr_level
+mr_rate
+node
+rate
+type
−scenario_stress
−scenario_mc
−shift_type
−timestep_mc
disp
getValue
get
set

See the class documentation in the next chapter for further information.

3.2 Implementation workflow

3.2.1 Overview

The following enumeration gives an overview of the main script *octarisk.m* (See Section 4.25 [octarisk], page 34 for details):

1. DEFINITION OF VARIABLES
   1. general variables
   2. VAR specific variables
2. INPUT
   1. Processing Instruments data
   2. Processing Riskfactor data
   3. Processing Positions data
   4. Processing Stresstest data
   5. Processing Market data
3. CALCULATION
   1. Model Riskfactor Scenario Generation
      - Load input correlation matrix
      - Get distribution parameters from riskfactors
      - call MC scenario generations
   2. Monte Carlo Riskfactor Simulation for all timesteps
   3. Take risk factor stress scenarios from stressdefinition
   4. Process yield curves and volatility surface
   5. Update market data objects with risk factor shocks
   6. Full Valuation of all Instruments in all MC and stress scenarios
   7. Portfolio Aggregation
      - loop over all portfolios / positions
      - VaR Calculation
        - sort arrays
        - Get Value of confidence scenario
        - make vector with Harrel-Davis Weights
      - Calculate Expected Shortfall
   8. Print Report including position VaRs
   9. Plotting
4. HELPER FUNCTIONS

3.3 Input files

In the following, all required input files are introduced. The basic file format for instruments, positions, stresstests, risk factors and market data objects is comma separated with a variable number of header attributes. Each of these input files is further split into different sub types. Each sub type has his own header, which is directly followed by all entries belonging to this sub type. Each Header line is introduced by the string **Header** and without any space followed by the **SUBTYPE** in capital letters (e.g. **HeaderFRB** for fixed rate bonds of the instrument class). Each header attribute contains the name of the header and the type of the data: **NameTYPE**. There exist exactly four different attribut types:

- *CHAR*: any character combination without commas. For definition of lists (e.g. several riskfactors and weights), also type CHAR is used. The list entries are separated by `|` (pipe symbol): `365|730|1095|3650|7300` can be used for a definition of curve nodes.
- *DATE*: date in the format *DD-MMM-YYYY* (e.g. 29-Apr-2016) for the use of maturity dates or issue dates.
- *BOOL*: boolean variable. Either *1* or *0* or *TRUE* or *FALSE*
- *NMBR*: numeric variable (can also be complex). Can be an integer or float with double precision.

Empty attribute values (e.g. *ValueA,,1234*) are ignored during parsing of the input files.

3.3.1 Risk factors

The risk factors input file contains all risk factors which are modeled by stochastic processes. The shocks of these risk factors are then used as input to the calculation of the scenario dependent index, curve, volatility and instrument values which have these risk factors as attached risk drivers.
The columns of the risk factors file consist of the following entries:

- *HeaderRISKFACTOR*: Introduction of risk factors
- *nameCHAR*: Name of the riskfactors, this follows the convention RF_TYPE_XYZ (string).
- *idCHAR*: Unique ID of the risk factors. To keep it simple, just take name (string).
- *typeCHAR*: Risk factor types follow typical asset class conventions (string). These types are explained in Section 2.1 [Introducing market risk], page 4.
- *descriptionCHAR*: A short description of the risk factor. String maximum length of 255 characters.
- *modelCHAR*: Model ID of the underlying stochastic process (string). See section Models for further explanation.
- *meanNMBR*: Mean of the stochastic process used in scenario generation
- *stdNMBR*: Standard deviation (expected *annualized* volatility)
- *skewNMBR*: Skewness
- *kurtNMBR*: Kurtosis
- *value_baseNMBR*: Start value for the mean reversion (e.g. Ornstein-Uhlenbeck or square root diffusion) processes
- *mr_levelNMBR*: Mean reversion level

- *mr_rateNMBR*: Mean reversion rate
- *nodeNMBR*: Comma separated parameters used for modeling the stochastic process.

An example of the input file is given:

```
HeaderRISKFACTOR,nameCHAR,idCHAR,typeCHAR,descriptionCHAR,modelCHAR,meanNMBR, ...
   ... stdNMBR,skewNMBR,kurtNMBR,value_baseNMBR,mr_levelNMBR,mr_rateNMBR,nodeNMBR
Item,RF_EQ_DE,RF_EQ_DE,RF_EQ,Equity Germany,GBM,0,0.18,-0.5,5,9820,,,
Item,RF_EQ_EUR,RF_EQ_EUR,RF_EQ,Equity Euro,GBM,0,0.18,-0.5,5,,,,
Item,RF_FX_EURUSD,RF_FX_EURUSD,RF_FX,FX EUR USD,SRD,0,0.08,0,3,1.09,1.2,0.001,
Item,RF_VOLA_EQ_DE,RF_VOLA_EQ_DE,RF_VOLA,Impl Vol Ger,SRD,0,0.1,0,3,0.31,0.21,0.02,
Item,RF_IR_EUR_1Y,RF_IR_EUR_1Y,RF_IR,IR EUR 1year,BM,0,0.0011,0,3,0.0008,,,365
Item,RF_IR_EUR_10Y,RF_IR_EUR_10Y,RF_IR,IR EUR 10year,BM,0,0.0032,0,3,0.0026,,,3650
Item,RF_IR_EUR_20Y,RF_IR_EUR_20Y,RF_IR,IR EUR 20year,BM,0,0.006,0,3,0.015,,,7300
Item,RF_VOLA_COM_GOLD,RF_VOLA_COM_GOLD,RF_VOLA,VolGold,SRD,0,0.1,0,3,0.15,0.16,0.03,
Item,RF_SPPR_EUR_HY_5Y,RF_SPR_EUR_HY_5Y,RF_SPR,HY,BM,0,0.083,0.24,10,0.05,,,1825
```

3.3.2 Positions

The positions input file contains all portfolios and positions. Positions must point to instruments which are defined in the instruments input file. Both portfolios and positions are stored to structures. Thus, additional columns can be appended, which could then be used as positional attributes.
The columns of the portfolio contain the following characteristics :

- *HeaderPORTFOLIO*: Indicating a new header specifying a portfolio
- *idCHAR*: Unique ID of the portfolio. Used in the filename of the report
- *nameCHAR*: Portfolio name
- *descriptionCHAR*: Description of the portfolio (optional)
- *HeaderPOSITION*: Indicating a new header specifying a position
- *port_idCHAR*: ID of the portfolio, where the position belongs to. Must be valid portfolio ID.
- *idCHAR*: ID of the instrument
- *quantityNMBR*: quantity of the instrument in the portfolio

Example definitions for some positions (positive quantity: long position, negative quantity: short position) in two portfolios:

```
HeaderPORTFOLIO,idCHAR,nameCHAR,descriptionCHAR
Item,FUND_AAA,Global Diversified,Global diversified test portfolio
Item,FUND_BBB,Global Derivatives,Global derivatives test portfolio
HeaderPOSITION,port_idCHAR,idCHAR,quantityNMBR
Item,FUND_AAA,A0RFFT,65
Item,FUND_AAA,A1JB4Q,179
Item,FUND_AAA,A1J7CK,135
Item,FUND_AAA,A1YC04,20
Item,FUND_AAA,BTCOIN,1.98
Item,FUND_AAA,ODAXC20160318,-0.01
Item,FUND_AAA,EQFORW01,1
Item,FUND_AAA,SYNTH01,0.1
Item,FUND_AAA,CASH_EUR,1000
Item,FUND_BBB,A0LGQL,723
Item,FUND_BBB,ODAXC20160318,-0.1
Item,FUND_BBB,EQFORW01,1
Item,FUND_BBB,SYNTH01,1
Item,FUND_BBB,CASH_EUR,1000
```

3.3.3 Instruments

The instruments input file contains the specifications of all instruments which are priced during instrument valuation. The instrument universe is split into different product types. Each of the product type has his own file header, reflecting the huge differences in instrument specification.
The basic header attributes, which all instruments have in common, are given. For detailed information see the class diagram (See ⟨undefined⟩ [Class diagram], page ⟨undefined⟩).

- *HeaderXXYYZ*: Product type specific header attributes, e.g. CASH, FRB, FRN, SENSI, SYNTH, OPT, FWD, SWAPT, ...
- *nameCHAR*: Name of the instrument which will be used for the reports.
- *idCHAR*: Unique ID of the instrument. Used as a reference in position input file. Is used as default aggregation key for reporting.
- *value_baseNMBR*: Actual market value of the instrument.
- *typeCHAR*: Instrument type specifying the sub type inside the specified instrument class (e.g. EQFWD for equity forward or OPT_EUR_C for an European call option).
- *descriptionCHAR*: A short description of the instrument. Maximum length of 255 characters.
- *currencyCHAR*: Currency of the instrument. If no appropriate currency risk factor is defined, they are mapped to EUR. Is used as default aggregation key for reporting.
- *asset_classCHAR*: Instrument asset class. Is used as default aggregation key for reporting.

Example definitions for cash instruments:

```
HeaderCASH,nameCHAR,idCHAR,value_baseNMBR,typeCHAR,descriptionCHAR,currencyCHAR,asset_classCHAR
Item,Cash Account EUR,CASH_EUR,1,CASH,EUR Cash Account,EUR,cash
```

3.3.4 Stress tests

The stress test input file contains the definition of all stress test. Each stress test describes the behavior of one or more risk factor in a particular scenario. The risk factor shock values are directly applied to all risk factor IDs which are selected through the regular expression. The columns of the stress test file consists of following entries:

qitem *HeaderSTRESSTESTS*: indicates stress tests

- *idCHAR*: Unique ID of the stresstest.
- *nameCHAR*: Name of the stresstest, used in reporting.
- *risktypeCHAR*: Pipe `|` separated IDs of risk types OR risk factors. Each risktype is treated as a regular expression applied to all risk factor IDs. Therefore it is e.g. possible to shock all high yield spread curves, regardless of their currency: *RF_SPREAD_.*_HY* or all EUR curves, regardless of rating: *IR_EUR* in the particular scenario. If a whole risk factor ID is given (e.g. RF_FX_EURUSD or RF_IR_EUR_5Y), only the single risk factors are shocked.
- *shiftvalueCHAR*: The value of the applied shocks. Relative shocks are given in decimals (e.g. -0.5 for a -50% down shock). Absolute shocks to interest rate nodes are given in basis points (e.g. 100 means a 100bp upshock).
- *shifttypeCHAR*: Integer 0 or 1 defining the shock type. 1 means relative shock (multiplied by risk factor base value), 0 means absolute shock (added to risk factor base value). Inside one stress test all curves nodes requires to have the same shift type.

An example for possible stress test definitions are given. E.g. the asian currency crisis stress test applies a relative down shock of 50% to all Emerging market equity risk factors and shocks all exchange rates (relative to EUR) by 10%:

```
#Stresstests Specifications
HeaderSTRESSTESTS,idCHAR,nameCHAR,risktypeCHAR,shiftvalueCHAR,shifttypeCHAR
Item,IR_100bpDOWN,IR - 100bp,RF_IR,-100,0
Item,IR_100bpUP,IR + 100bp,RF_IR,100,0
Item,SPREAD_100bpDOWN,Spread - 100bp,RF_SPREAD,-100,0
Item,SPREAD_100bpUP,Spread + 100bp,RF_SPREAD,100,0
Item,EQ_30DOWN,Equity - 3000bp,RF_EQ,-0.3,1
Item,EQ_30UP,Equity + 3000bp,RF_EQ,0.3,1
Item,VOLA_100UP,VOLA + 10000bp,RF_VOLA,1,1
Item,RE_30DOWN,Real Estate - 3000bp,RF_RE,-0.3,1
Item,RE_30UP,Real Estate + 3000bp,RF_RE,0.3,1
Item,COM_30DOWN,Commodity - 3000bp,RF_COM,-0.3,1
Item,COM_30UP,Commodity + 3000bp,RF_COM,0.3,1
Item,EUR_USD_FX_30DOWN,FX - 3000bp,RF_FX,-0.3,1
Item,EUR_USD_FX_30UP,FX + 3000bp,RF_FX,0.3,1
Item,Terror2001-9-11,Terror 2001-9-11,RF_EQ|RF_RE|RF_VOLA_EQ,-0.15|-0.15|0.3,1|1|1
Item,DotCom2000,DotCom bubble 2000,RF_EQ|RF_VOLA_EQ|RF_SPREAD,-0.35|0.5|150,1|1|0
Item,AsianCurrCrisis1997,Asian Currency Crisis 1997,RF_EQ_EM|RF_FX,-0.5|-0.1,1|1
```

New stress tests can be easily appended to the specification file.

3.3.5 Volatility surface

For all options and swaptions, the implied volatility is necessary to calculate the derivative theoretical value. In order to feed the implied volatility into the system, a term / moneyness surface has to be specified for index instruments and a underlying tenor / term / moneyness for IR instruments in a separate file. For all underlying index risk factors the impl. volatility data file has to be named like *vol_index_RF_XX_YY.dat* and *vol_ir_RF_XX_YY.dat* for all interest rate risk factor. The risk factor ID will be used to automatically identify the appropriate file. The structure of the file is a linearized version of the volatility surface (for term / moneyness instruments):

```
% term moneyness implied_vola
30 1.2 0.4
30 1.0 0.35
30 0.8 0.3
90 1.2 0.5
90 1.0 0.45
90 0.8 0.4
```

and the volatility cube for tenor term moneyness for interest rate instruments:

```
#tenor term moneyness implied_vola
365.00000 365.00000 1.00000 0.50000
365.00000 730.00000 1.00000 0.40000
365.00000 1095.00000 1.00000 0.30000
730.00000 730.00000 1.00000 0.35000
730.00000 365.00000 1.00000 0.30000
730.00000 1095.00000 1.00000 0.35000
365.00000 365.00000 1.25 0.50000
365.00000 730.00000 1.25 0.50000
365.00000 1095.00000 1.25 0.30000
730.00000 730.00000 1.25 0.35000
730.00000 365.00000 1.25 0.30000
730.00000 1095.00000 1.25 0.35000
```

All tenor and term values are given in days from valuation date.

During the full valuation approach the moneyness is a function of the underlying risk factor spot price over rate and the constant strike price over rate. A linear interpolation for the moneyness and a nearest neighbour mapping for tenors terms and constant extrapolation will be performed to calculate the new scenario dependent implied volatililty. Since the at-the-money volatility is itself a risk factor (modeled as a factor), the interpolated volatility will be adjusted by this scenario dependent factor. This process combines the conservation of volatility surface or cubes shape with the single factor stochastic modeling of the at-the-money volatility. Furthermore, it is also possible to generate a file with just one constant volatility.

3.3.6 Marketdata objects file

Market data objects store all relevant objects for full valuation instrument pricing. These objects can be interest and spread curves, market indizes and exchange rates. The base values of these objects are then shocked by scenario dependent values, which were calculated for the attached risk factors. The market data objects are specified in a separate file following the same conventions as for instruments:

qitem *HeaderCURVE*: market object specific Header classification for curves (e.g. HeaderCURVE or HeaderINDEX)

- *idCHAR*: Unique ID of the market curve. Note, that the ID of the market curve must have an attached risk factor starting with *RF_* followed by the market curve ID. Otherwise, an automatic mapping is not possible.

- *nameCHAR*: Name of the market curve.
- *typeCHAR*: Type of the curve (e.g. Spread Curve or Discount Curve)
- *descriptionCHAR*: Description of the object.
- *method_interpolationCHAR*: Interpolation method for the market curve (e.g. linear or monotone-convex). This interpolation method can be different to the interpolation method from the attached risk factor. For all nodes of the market curve the relative of absolute scenario dependent shock values (Derived from the attached risk factor curve) is interpolated and then applied to the market curve node.
- *nodesCHAR*: Pipe separated nodes of the market curve in days from the valuation date.
- *rates_baseCHAR*: Pipe separated rates of the market curve. One rate needed for each specified node.

qitem *HeaderINDEX*: market object specific Header classification for indizes and exchange rates. Basically, all market objects with exactly one scenario dependent value can be stored here.

- *idCHAR*: Unique ID of the market index. Note, that the ID of the market index must have an attached risk factor starting with *RF_* followed by the market index ID. Otherwise, an automatic mapping is not possible.
- *nameCHAR*: Name of the market index.
- *typeCHAR*: Type of the index (e.g. Equity Index, Commodity Index, Exchange Rate)
- *currencyCHARdescriptionCHAR*: Currency of the object.
- *descriptionCHAR*: Description of the object
- *value_baseNMBR*: Market value of the index.

```
HeaderCURVE,idCHAR,nameCHAR,typeCHAR,descriptionCHAR,method_interpolationCHAR,nodesCHAR,rates_
Curve,IR_EUR,IR_EUR,Discount Curve,EUR-SWAP Curve,monotone-convex,365|1095|1825|3650|7300|1095
0.00519251|-0.00508595|-0.00367762|0.00185694|0.00776253|0.00999986|0.00123
Curve,IR_USD,IR_USD,Discount Curve,USD-SWAP Curve,monotone-convex,365|1095|1825|3650|7300|1095
Curve,SPREAD_EUR_HY,SPREAD_EUR_HY,Spread Curve,SPREAD_EUR_High Yield Curve,linear,1825,0.02
HeaderINDEX,idCHAR,nameCHAR,typeCHAR,currencyCHAR,descriptionCHAR,value_baseNMBR
Index,EQ_DE,DAX30,Equity Index,EUR,DAX Equity Index German Blue Chips,9820
Index,COM_GOLD,Gold,Commodity Index,USD,Gold 1 Ounce USD,1073
Index,FX_EURUSD,EUR_USD,Exchange Rate,EUR,EUR USD Exchange rate,1.08
```

These two market objects (indizes and curves) reflect market objects with either just one scenario dependent value (a scalar like for indizes) or a certain constant number of dependent values per scenario (like a curve). For these objects it is possible to specify an arbitrary number of risk factors in order to get the most granular scenario dependent behaviour. For convenience reasons and because of the increased memory consumption, indizes and curves are up to now the only possible market objects. For volatility surface or cubes it is only possible to specify risk factors with just one scenario dependent value (at-the-money volatility). The market object (the volatility cube) is then offsetted by a constant value in each scenario, thus preserving the base value volatility smile. It is therefore not possible to model a scenario dependent smile.

3.3.7 Correlation matrix

The correlation file is a full matrix containing the correlations between all risk factors. The first line can be a header giving the IDs of the risk factors, but this information is not used during parsing. Therefore the dimension of the correlation matrix is n x n, where n must be the number of defined risk factors (otherwise an exception will be raised). The order of the risk factors in the correlation matrix must be reflected in the order of the risk factor input file. The correlation matrix undergoes a Cholesky decomposition to generate correlated random variables with a copula approach. The four moments of the marginal risk factors distributions are directly given in the risk factor input file. If the correlation matrix is not positive semi-definite, all negative eigenvalues are set to slightly positive numbers, thus leading to positive semi-definiteness. Be aware, that the correlations may change. Further statistics are automatically calculated if this transformation procedure is necessary.

3.4 Output files

Overview of report summary and graphics.

3.4.1 VAR Report

For each fund, a VAR report is generated. The report shall give an overview of total risk measure, allow a break down to position level and estimate the diversification effect. An example for the report looks like:

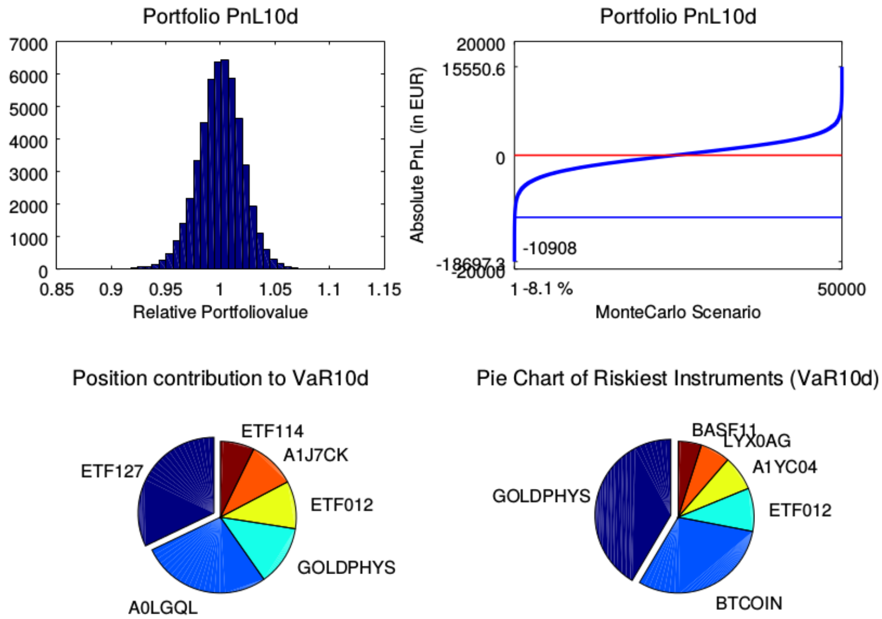
```
=== Value-At-Risk Report for Portfolio 1 ===
VaR calculated 99pct Confidence Intervall:
Number of Monte Carlo Scenarios: 500000
Confidence Scenarionummer: 5000
Valuation Date: 09-Oct-2015
VaR on Positional Level:
|VaR 1D for Position   |iShares JPMorgan $ EM|A0RFFT| = | 618.16 EUR|
|VaR 250D for Position |iShares JPMorgan $ EM|A0RFFT| = | 1755.21 EUR|
...
=== Total Portfolio VaR ===
|Portfolio VaR   1D@99Pct|   |    -1.32%|
|Portfolio VaR   1D@99Pct|   |  1779.97 EUR|
|Portfolio VaR 250D@99Pct|   |   -19.34%|
|Portfolio VaR 250D@99Pct|   | 25983.52 EUR|

|Expected Shortfall 1D@99Pct|   |    -1.51%|
|Expected Shortfall 1D@99Pct|   |  2027.10 EUR|
|Expected Shortfall 250D@99Pct|   |   -21.58%|
|Expected Shortfall 250D@99Pct|   | 28996.72 EUR|
```

The unique field separator '|' allows efficient use of the data in LaTeX based report files.

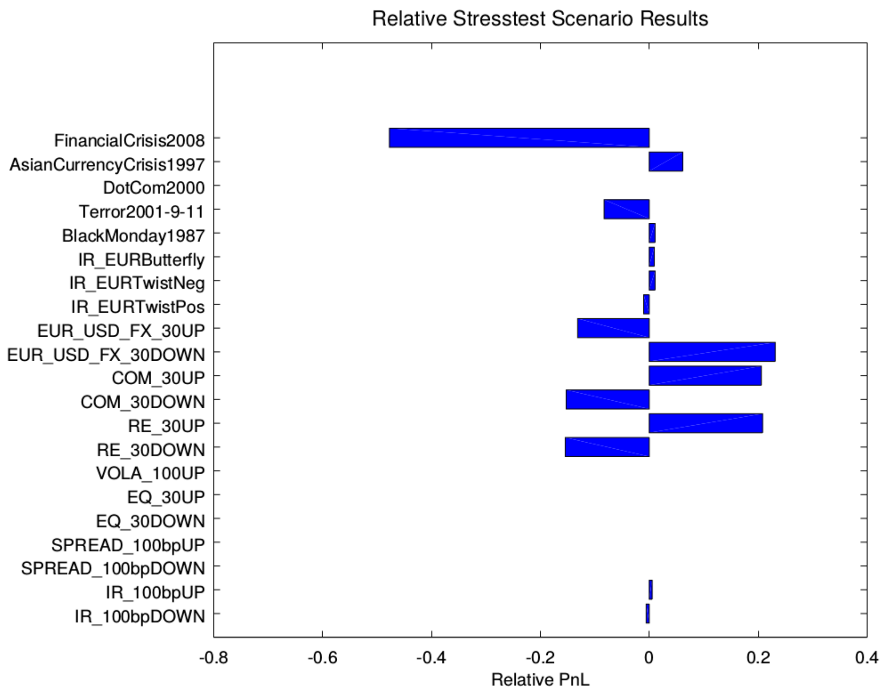
3.4.2 Overview images

Additionally to the printed report, overview images of the profit and loss distribution, both sorted and histogram as well as the riskiest instruments and positions are plotted.



Portfolio Value-at-Risk at the 99.9% confidence level on 10 day time horizon. There are shown four different results: On the top left corner a histogram of the profit and loss distribution is presented for the whole portfolio. On the x-axis, the relative portfolio value is given. On the top right corner the P'n'L distribution is shown as sorted absolute profits or losses for each of the MC scenarios. The red base line marks the base scenario. The blue line indicates the VAR scenario number, where 99.9% of all profits and losses are shown on the right side. Losses greater than the VAR occur in 0.1% of all cases. The two lower charts show the VAR contribution of the six riskiest positions (left chart) or instruments (right chart). The positional view is determined by weighing the instruments with their position size in the portfolio.

Moreover, stress test results are plotted in a bar chart, indicating the relative profit or loss of the fund in each stress test scenario:


Relative Stresstest Scenario Results
FinancialCrisis2008
AsianCurrencyCrisis1997
DotCom2000
Terror2001-9-11
BlackMonday1987
IR_EURButterfly
IR_EURTwistNeg
IR_EURTwistPos
EUR_USD_FX_30UP
EUR_USD_FX_30DOWN
COM_30UP
COM_30DOWN
RE_30UP
RE_30DOWN
VOLA_100UP
EQ_30UP
EQ_30DOWN
SPREAD_100bpUP
SPREAD_100bpDOWN
IR_100bpUP
IR_100bpDOWN
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
Relative PnL

4 Octave Functions and Scripts

In the following sections you find the Octave function texinfo.

4.1 calibrate_evt_gpd

`[chi sigma u] = calibrate_evt_gpd(v)` [Function File]

Calibrate sorted losses of historic or MC portfolio values to a generalized pareto distribution and returns chi, sigma and u as parameters for further VAR and ES calculation.

Implementation according to *Risk Management and Financial Institutions* by John C. Hull, 4th edition, Wiley 2015, section 13.6, page 292ff.

Explanation of Parameters:

- *v*: INPUT: sorted profit and loss distribution of all required tail events (1xN vector)
- *chi*: OUTPUT: Generalized Pareto distribution: shape parameter (scalar)
- *sigma*: OUTPUT: scale parameter (scalar)
- *u*: OUTPUT: location parameter(scalar)

4.2 calibrate_option_bs

`[vola_spread] = calibrate_option_bs(putcallflag,S,X,T,rf,sigma,multiplicator,market_value)` [Function File]

Calibrate the implied volatility spread for Eeuropean options according to Black-Scholes valuation formula.

4.3 calibrate_option_willowtree

`[vola_spread] = calibrate_option_willowtree(putcallflag,americanflag,S,X,T,rf,sigma,divyield,st` [Function File]

Calibrate the implied volatility spread for American options according to Willowtree valuation formula.

4.4 calibrate_soy_sqp

`[vola_spread] = calibrate_soy_sqp(valuation_date,tmp_cashflow_dates,tmp_cashflow_values,tmp_act` [Function File]

Calibrate the spread over yield according to given cashflows discounted on an appropriate yield curve.

4.5 calibrate_swaption

`[vola_spread] = calibrate_swaption(PayerReceiverFlag,F,X,T,r,sigma,m,tau,multiplicator,market_v` [Function File]

Calibrate the implied volatility spread for European swaptions according to Black76 or Bachelier (default) valuation model. In extreme out-of-the-money scenarios there could be no solution in changing the volatility. But we dont care in this case at all, since the influence of the volatility is neglectable in this extreme cases.

4.6 calibrate_yield_to_maturity

`[vola_spread] = calibrate_yield_to_maturity(valuation_date,tmp_cashflow_dates,tmp_cashflow_valu` [Function File]

Calibrate the yield to maturity according to given cashflows.

4.7 correct_correlation_matrix

`[A_scaled pos_sem_def_bool] = correct_correlation_matrix(M)` [Function File]

Return a positive semi-definite matrix *A_scaled* to a given input matrix *M*. This function tests for indefiniteness of the input matrix and eventuallry adjusts negative Eivenvalues to 0 or slightly positive values via some iteration steps.
Reference: 'Implementing Value at Risk', Best, Philip W., 1998.

4.8 discount_factor

`df = discount_factor (d1, d2, rate, comp_type, basis, comp_freq)` [Function File]

Compute the discount factor for a specific time period, compounding type, day count basis and compounding frequency.

Input and output variables:

- *d1*: number of days until first date (scalar)
- *d2*: number of days until second date (scalar)
- *rate*: interest rate between first and second date (scalar)
- *comp_type*: compounding type: [simple, simp, disc, discrete, cont, continuous] (string)
- *basis*: day-count basis (scalar)
  - *0* = actual/actual
  - *1* = 30/360 SIA (default)
  - *2* = act/360
  - *3* = act/365
  - *4* = 30/360 PSA
  - *5* = 30/360 ISDA
  - *6* = 30/360 European
  - *7* = act/365 Japanese
  - *8* = act/act ISMA
  - *9* = act/360 ISMA
  - *10* = act/365 ISMA
  - *11* = 30/360E (ISMA)
- *comp_freq*: 1,2,4,12,52,365 or [daily,weekly,monthly,quarter,semi-annual,annual] (scalar or string)

- *df*: OUTPUT: discount factor (scalar)

See also: timefactor.

4.9 doc_instrument

`object = Instrument (name,id,description,type,currency,base_value,asset_class,valuation_date)` [Function File]

Instrument Superclass Inputs:

- *name* (string): Name of object
- *id* (string): Id of object
- *description* (string): Description of object
- *type* (string): instrument type in list [cash, bond, debt, forward, option, sensitivity, synthetic]
- *currency* (string): ISO code of currency
- *base_value* (float): Actual base (spot) value of object
- *asset_class* (sring): Instrument asset class
- *valuation_date* (datenum): serial day number from Jan 1, 0000 defined as day 1.

The constructor of the instrument class constructs an object with the following properties and inherits them to all sub classes:

- name: Name of object
- id: Id of object
- description: Description of object
- value_base: Actual base (spot) value of object
- currency: ISO code of currency
- asset_class: Instrument asset class
- type: Type of Instrument class (Bond,Forward,...)
- valuation_date: date format DD-MMM-YYYY
- value_stress: Vector with values under stress scenarios
- value_mc: Matrix with values under MC scenarios (values per timestep per column)
- timestep_mc: MC timestep per column (cell string)

value = Instrument.getValue (*base*,*stress*,*mc_timestep*) Superclass Method getValue Return the scenario (shock) value for an instrument object. Specify the desired return values with a property parameter. If the second argument abs is set, the absolut scenario value is calculated from scenario shocks and the risk factor start value.
Timestep properties:

- base: return base value
- stress: return stress values
- 1d: return MC timestep

boolean = Instrument.isProp (property) Instrument Method isProp
Query all properties from the Instrument Superclass and sub classes and returns 1 in case of a valid property.

See also: Instrument.

4.10 doc_riskfactor

object = `Riskfactor` () [Function File]
object = `Riskfactor` (*name*,*id*,*type*,*description*,*model*,*parameters*) [Function File]

Construct risk factor object. Riskfactor Class Inputs:

- *name* (string): Name of object
- *id* (string): Id of object
- *type* (string): risk factor type
- *description* (string): Description of object
- *model* (string): statistical model in list [GBM,BM,SRD,OU]
- *parameters* (vector): vector with values [mean,std,skew,kurt,start_value,mr_level,mr_rate,node,rate]

If no input arguments are provided, a dummy IR risk factor object is generated.
The constructor of the risk factor class constructs an object with the following properties:
Class properties:

- name: Name of object
- id: Id of object
- description: Description of object
- type: risk factor type
- model: risk factor model
- mean: first moment of risk factor distribution
- std: second moment of risk factor distribution
- skew: third moment of risk factor distribution
- kurt: fourth moment of risk factor distribution
- start_value: Actual spot value of object
- mr_level: In case of mean reverting model this is the mean reversion level
- mr_rate: In case of mean reverting model this is the mean reversion rate
- node: In case of a interest rate or spread risk factor this is the term node
- rate: In case of a interest rate or spread risk factor this is the term rate at the node
- scenario_stress: Vector with values of stress scenarios
- scenario_mc: Matrix with risk factor scenario values (values per timestep per column)

- timestep_mc: MC timestep per column (cell string)

Function File *property_value* = Riskfactor.getValue ((*base,stress,mc_timestep*), 'abs')
Riskfactor Method getValue
Return the value for a risk factor object. Specify the desired return values with a property parameter. If the second argument abs is set, the absolut scenario value is calculated from scenario shocks and the risk factor start value.
Timestep properties:

- base: return base value
- stress: return stress values
- any regular MC timestep (e.g. '1d'): return scenario (shock) values at MC timestep

property_value = Riskfactor.get (property) *object* = Riskfactor.set (property, value)
Riskfactor Methods get / set
Get / set methods for retrieving or setting risk factor properties.

See also: Instrument.

4.11 estimate_parameter

`estimate_parameter()` [Function File]

Calculate statistics for historic time series. This script is not used in octarisk process, but it should simplify the parameter estimation for input parameters required for risk factor definitions.
The time series file (columns: risk factors, rows: historic values) will be generated by an python script (to be done).
The following statistic parameter are calculated: mean, volatility, skewness, kurtosis for simple, geometric and lognormal distributed value. Moreover mean reversion rates and levels are calculated assuming ergodic and regression methods.

4.12 get_documentation

`get_documentation(`*`type,path_documentation`*`))` [Function File]

Print documentation for all specified Octave functions. This script allows you to choose functions, for which the documentation is extracted from the function headers and printed to a file 'functions.texi', 'functionname.html' or to standard output if a specific format (texinfo, html, txt) is set. The path to all files has to be set in the variable path_documentation.

4.13 get_forward_rate

`forward_rate``= get_forward_rate(`*`nodes`*`,` *`rates`*, *`days_to_t1`*, *`days_to_t2`*, *`comp_type`*, *`interp_method`*`)` [Function File]

Compute the forward rate calculated from interpolated rates from a zero coupon yield curve. CAUTION: the forward rate is floored to 0.000001. Explanation of

Input Parameters:

- *nodes*: is a 1xN vector with all timesteps of the given curve
- *rates*: is MxN matrix with curve rates defined in columns. Each row contains a specific scenario with different curve structure
- *days_to_t1*: is a scalar, specifiying term1 in days
- *days_to_t2*: is a scalar, specifiying term2 in days after term1
- *comp_type*: (optional) specifies compounding rule (simple, discrete, continuous (defaults to 'cont')).
- *interp_method*: (optional) specifies interpolation method for retrieving interest rates (defaults to 'linear').

See also: interpolate_curve.

4.14 get_marginal_distr_pearson

`[r type ] = get_marginal_distr_pearson (mu, sigma, skew, kurt, Z)` [Function File]

Compute a marginal distribution for given set of uniform random variables with given mean, standard deviation skewness and kurtosis. The mapping is done via the Pearson distribution family.
The implementation is based on the R package 'PearsonDS: Pearson Distribution System' and the function 'pearsonFitM' by
Martin Becker and Stefan Kloessner (2013)
R package version 0.97.
URL http://CRAN.R-project.org/package=PearsonDS
licensed under the GPL >= 2.0
Input and output variables:

- *mu*: mean of marginal distribution (scalar)
- *sigma*: standard deviation of marginal distribution (scalar)
- *skew*: skewness of marginal distribution (scalar)
- *kurt*: kurtosis of marginal distribution (scalar)
- *Z*: uniform distributed random variables (Nx1 vector)
- *r*: OUTPUT: Nx1 vector with random variables distributed according to Pearson type (vector)
- *type*: OUTPUT: Pearson distribution type (I - VII) (scalar)

The marginal distribution type is chosen according to the input parameters out of the Pearson Type I-VII distribution family:

- *Type 0* = normal distribution
- *Type I* = generalization of beta distribution
- *Type II* = symmetric beta distribution
- *Type III* = gamma or chi-squared distribution

- *Type IV* = special distribution, not related to any other distribution
- *Type V* = inverse gamma distribution
- *Type VI* = beta-prime or F distribution
- *Type VII* = Student's t distribution

See also: discount_factor.

4.15 harrell_davis_weight

`harrell_davis_weight (`*`scenarios`*`, `*`observation`*`, `*`alpha`*`)` [Function File]

Compute the Harrell-Davis (1982) quantile estimator and jacknife standard errors of quantiles. The quantile estimator is a weighted linear combination or order statistics in which the order statistics used in traditional nonparametric quantile estimators are given the greatest weight. In small samples the H-D estimator is more efficient than traditional ones, and the two methods are asymptotically equivalent. The H-D estimator is the limit of a bootstrap average as the number of bootstrap resamples becomes infinitely large.

4.16 interpolate_curve

`interpolate_curve (`*`nodes`*`, `*`rates`*`, `*`timestep`*`)` [Function File]
`interpolate_curve (`*`nodes`*`, `*`rates`*`, `*`timestep`*`, `*`ufr`*`, `*`alpha`*`)` [Function File]

Calculate an interpolated return on a curve for a given timestep Supported methods are: linear (default), moneymarket, exponential, loglinear, spline, smith-wilson or monotone-convex. A constant extrapolation is assumed, except for smith-wilson, where the ultimate forward rate will be reached proportional to reversion speed alpha. For all methods except splines a fast taylormade algorithm is used. For splines see Octave function interp1 for more details. Explanation of Input Parameters of the interpolation curve function:

- *nodes*: is a 1xN vector with all timesteps of the given curve
- *rates*: is MxN matrix with curve rates per timestep defined in columns. Each row contains a specific scenario with different curve structure
- *timestep*: is a scalar, specifiying the interpolated timestep on vector nodes
- *ufr*: OPTIONAL: (only used for smith-wilson): ultimate forward rate (default: last liquid point)
- *alpha*: OPTIONAL: (only used for smith-wilson): reversion speed to ultimate forward rate (default: 0.1)

See also: interp1, interp2, interp3, interpn.

4.17 load_instruments

`[instrument_struct id_failed_cell] =` [Function File]
`load_instruments(instrument_struct,valuation_date,path_instruments,file_ins`

Load data from instrument specification file and generate objects with parsed data. Store all objects in provided instrument struct and return the final struct and a cell containing the failed instrument ids.

4.18 load_positions

`[portfolio_struct id_failed_cell] =` [Function File]
`load_positions(portfolio_struct, valuation_date,path_positions,file_positions,path_output,path_archive,tmp_timestamp`

Load data from position specification file and generate objects with parsed data. Store all objects in provided position struct and return the final struct and a cell containing the failed position ids.

4.19 load_riskfactor_scenarios

`[riskfactor_struct rf_failed_cell] =` [Function File]
`load_riskfactor_scenarios(riskfactor_struct, M_struct,mc_timesteps,mc_timestep_days)`

Generate MC scenario shock values for risk factor curve objects. Store all MC scenario shock values in provided struct and return the final struct and a cell containing all failed risk factor ids.

4.20 load_riskfactor_stresses

`[riskfactor_struct rf_failed_cell] =` [Function File]
`load_riskfactor_stresses(riskfactor_struct, stresstest_struct)`

Generate stresses for risk factor curve objects. Store all stresses in provided struct and return the final struct and a cell containing all failed risk factor ids.

4.21 load_riskfactors

`[riskfactor_struct id_failed_cell] =` [Function File]
`load_riskfactors(riskfactor_struct,valuation_date,path_riskfactors,file_ris`

Load data from riskfactor specification file and generate objects with parsed data. Store all objects in provided riskfactor struct and return the final struct and a cell containing the failed riskfactor ids.

4.22 load_stresstests

[*portfolio_struct id_failed_cell*] = load_stresstests(*portfolio_struct*, *valuation_date*,*path_stresstests*,*file_stresstests*,*path_output*,*path_archive*,*tmp_times* [Function File]

Load data from stresstest specification file and generate a struct with parsed data. Store all stresstests in provided struct and return the final struct and a cell containing the failed position ids.

4.23 load_volacubes

[*surface_struct vola_failed_cell*] = load_volacubes(*surface_struct*, *path_mktdata*,*input_filename_vola_index*,*input_filename_vola_ir*) [Function File]

Load data from mktdata volatility surfaces / cubes specification files and generate a struct with parsed data. Store all stresstests in provided struct and return the final struct and a cell containing the failed volatility ids.

4.24 load_yieldcurves

[*rf_ir_cur_cell curve_struct*] = load_volacubes(*curve_struct*, *riskfactor_struct*,*mc_timesteps*,*path_output*,*saving*) [Function File]

Generate curve objects from risk factor objects. Store all curves in provided struct and return the final struct and a cell containing all interest rate risk factor currency / ratings.

4.25 octarisk

octarisk (*path_working_folder*) [Function File]

Version: 0.0.0, 2015/11/24, Stefan Schloegl: initial version
0.0.1, 2015/12/16, Stefan Schloegl: added Willow Tree model for pricing american equity options,
added volatility surface model for term / moneyness structure of volatility for index vol
0.0.2, 2016/01/19, Stefan Schloegl: added new instrument types FRB, FRN, FAB, ZCB
added synthetic instruments (linear combinations of other instruments)
added equity forwards and Black-Karasinski stochastic process
0.0.3, 2016/02/05, Stefan Schloegl: added spread risk factors, general cash flow pricing
0.0.4, 2016/03/28, Stefan Schloegl: changed the implementation to object oriented approach (introduced instrument, risk factor classes)
0.1.0, 2016/04/02, Stefan Schloegl: added volatility cube model (Surface class) for tenor / term / moneyness structure of volatility for ir vol
0.2.0, 2016/04/19, Stefan Schloegl: improved the file interface and format for input files (market data, risk factors, instruments, positions, stresstests)

Calculate Monte-Carlo Value-at-Risk (VAR) and Expected Shortfall (ES) for instruments, positions and portfolios at a given confidence level on 1D and 250D time horizon with a full valuation approach.
See octarisk documentation for further information.

Input files in csv format:

- Instruments data: specification of instrument universe (name, id, market value, underlying risk factor, cash flows etc.)
- Riskfactors data: specification of risk factors (name, id, stochastic model, statistic parameters)
- Positions data: specification of portfolio and position data (portfolio id, instrument id, position size)
- Stresstest data: specification of stresstest risk factor shocks (stresstest name, risk factor shock values and types)
- Covariance matrix: covariance matrix of all risk factors
- Volatility surfaces (index volatility: term vs. moneyness, call moneyness spot / strike, linear interpolation and constant extrapolation)

Output data:

- portfolio report: instruments and position VAR and ES, diversification effects
- profit and loss distributions: plot of profit and loss histogramm and distribution, most important positions and instruments

Supported instrument types:

- equity (stocks and funds priced via multi-factor model and idiosyncratic risk)
- commodity (physical and funds priced via multi-factor model and idiosyncratic risk)
- real estate (stocks and funds priced via multi-factor model and idiosyncratic risk)
- custom cash flow instruments (NPV of all custom CFs)
- bond funds priced via duration-based sensitivity approach
- fixed rate bonds (NPV of all CFs)
- floating rate notes (scenario dependent cash flow values, NPV of all CFs)
- fixed amortizing bonds (either annuity bonds or amortizable bonds, NPV of all CFs)
- zero coupon bonds (NPV of notional)
- European equity options (Black-Scholes model)
- American equity options (Willow Tree model)
- European swaptions (Black76 and Bachelier model)
- Equity forward
- Synthetic instruments (linear combinations of other valuated instruments)

Supported stochastic processes for risk factors:

- Geometric Brownian Motion
- Black-Karasinski process
- Brownian Motion
- Ornstein-Uhlenbeck process
- Square-root diffusion process

Supported copulas for MC scenario generation:

- Gaussian copula
- t-copula with one parameter specification for common degrees of freedom

Further functionality will be implemented in the future (e.g. inflation linked instruments)

See also: option_willowtree, option_bs, harrell_davis_weight, swaption_black76, pricing_forward, rollout_cashflows, scenario_generation_MC.

4.26 option_bs

[*value delta gamma vega theta rho omega*] = option_bs (*CallPutFlag*, *S*, *X*, *T*, *r*, *sigma*, *divrate*) [Function File]

Compute the prices of european call or put options according to Black-Scholes valuation formula:

```
C(S,T) = N(d_1)*S - N(d_2)*X*exp(-rT)
P(S,T) = N(-d_2)*X*exp(-rT) - N(-d_1)*S
d1 = (log(S/X) + (r + 0.5*sigma^2)*T)/(sigma*sqrt(T))
d2 = d1 - sigma*sqrt(T)
```

The Greeks are also computed (delta, gamma, vega, theta, rho, omega) by their closed form solution.

Parallel computation for column vectors of S,X,r and sigma is possible.

Variables:

- *CallPutFlag*: Call: '1', Put: '0'
- *S*: stock price at time 0
- *X*: strike price
- *T*: time to maturity in days
- *r*: annual risk-free interest rate (continuously compounded)
- *sigma*: implied volatility of the stock price measured as annual standard deviation
- *divrate*: dividend rate p.a., continously compounded

See also: option_willowtree, swaption_black76.

4.27 option_willowtree

value = option_willowtree (*CallPutFlag*, *AmericanFlag*, *S*, *X*, *T*, *r*, *sigma*, *dividend*, *dk*) [Function File]

value = option_willowtree (*CallPutFlag*, *AmericanFlag*, *S*, *X*, *T*, *r*, *sigma*, *dividend*, *dk*, *nodes*) [Function File]

Computes the price of european or american equity options according to the willow tree model.

The willow tree approach provides a fast and accurate way of calculating option prices. Furthermore, massive parallelization due to litte memory consumption is possible. This implementation of the willow tree concept is based on following literature:

- 'Willow Tree', Andy C.T. Ho, Master thesis, May 2000
- 'Willow Power: Optimizing Derivative Pricing Trees', Michael Curran, ALGO RESEARCH QUARTERLY, Vol. 4, No. 4, December 2001

Efficient parallel computation for column vectors of S,X,r and sigma is possible (advantage: linear increase of calculation time in timesteps and nodes).

Runtime of parallel computations incl. tree transition optimization (360 days maturity, 5 day stepsize, 20 willow tree nodes) are performed (at 46 GFlops machine, 4 Gb Ram) in:

50 | 0.5s
500 | 0.5s
5000 | 1.1s
50000 | 9.0s
200000 | 32s

Example of an American Call Option with continuous dividends:
(365 days to maturity, vector with different spot prices and volatilities, strike = 8, r = 0.06, dividend = 0.05, timestep 5 days, 20 nodes): `option_willowtree(1,1,[7;8;9;7;8;9],8,365,0.06,[0.2;0.2;0.2;0.3;0.3;0.3],0.05,5,20)`

Variables:

- *CallPutFlag*: Call: '1', Put: '0'
- *AmericanFlag*: American option: '1', European Option: '0'
- *S*: stock price at time 0
- *X*: strike price
- *T*: time in days to maturity
- *r*: annual risk-free interest rate (continuously compounded, act/365)
- *sigma*: implied volatility of the stock price measured as annual standard deviation
- *dividend*: continuous dividend yield, act/365
- *dk*: size of timesteps for valuation points (optimal accuracy vs. runtime choice : 5 days timestep)
- *nodes*: number of nodes for willow tree setup. Number of nodes must be in list [10,15,20,30,40,50]. These vectors are optimized by Currans suggested Method to fulfill variance constraint (optimal accuracy vs. runtime choice: 20 nodes)

See also: option_binomial, option_bs, option_exotic_mc.

4.28 pricing_forward_oop

[*theo_value*] = pricing_forward_oop (*forward*, *discount_nodes*, *discount_rates*) [Function File]

Compute the theoretical value of equity and bond forwards.
In the 'oop' version no input data checks are performed.
Pre-requirements:

- installed octave financial package
- custom functions timefactor, discount_factor, interpolate_curve

Input and output variables:

- *forward*: Structure with relevant information for specification of the object forward:
  - forward.type Equity, Bond
  - forward.currency [optional] Currency (string, ISO code)
  - forward.maturity_date Maturity date of forward
  - forward.valuation_date [optional] Valuation date (default today)
  - forward.strike_price strike price (float). Can be a 1xN vector.
  - forward.underlying_price market price of underlying (float), in forward currency units. Can be a 1xN vector.
  - forward.storage_cost [optional] continuous storage cost p.a. (default 0)
  - forward.convenience_yield [optional] continuous convenience yield p.a. (default 0)
  - forward.dividend_yield [optional] continuous dividend yield p.a. (default 0)
  - forward.compounding_type [optional] compounding type [simple, discrete, continuous] (default 'disc')
  - forward.compounding_frequency [optional] compounding frequency [daily, weekly, monthly, semi-annual, annual] (default 'daily')
  - forward.day_count_convention [optional] day count convenction (e.g. 'act/act' or '30/360E') (default 'act/act')
- *discount_nodes*: tmp_nodes is a 1xN vector with all timesteps of the given curve
- *discount_rates*: tmp_rates is a MxN matrix with discount curve rates defined in columns. Each row contains a specific scenario with different curve structure
- *theo_value*: returs a 1xN vector with all forward values per scenario

See also: timefactor, discount_factor, interpolate_curve.

4.29 pricing_npv

[*npv MacDur*] = pricing_npv(*valuation_date*, *cashflow_dates*, *cashflow_values*,*spread_constant* ... ,*discount_nodes*, *discount_rates*, *spread_nodes*, *spread_rates*, *basis*, *comp_type*, *comp_freq*, *interp_discount*, *interp_spread*) [Function File]

Computes the net present value and Maccaulay Duration of a given cash flow pattern according to a given discount curve, spread curve and day count convention etc.
Pre-requirements:

- installed octave finance package
- custom functions timefactor, discount_factor, interpolate_curve

Input and output variables:

- *valuation_date*: Structure with relevant information for specification of the forward:

- *cashflow_dates*: cashflow_dates is a 1xN vector with all timesteps of the cash flow pattern
- *cashflow_values*: cashflow_values is a MxN matrix with cash flow pattern.
- *spread_constant*: a constant spread added to the total yield extracted from discount curve and spread curve (can be used to spread over yield)
- *discount_nodes*: tmp_nodes is a 1xN vector with all timesteps of the given curve
- *discount_rates*: tmp_rates is a MxN matrix with discount curve rates defined in columns. Each row contains a specific scenario with different curve structure
- *spread_nodes*: OPTIONAL: spread_nodes is a 1xN vector with all timesteps of the given spread curve
- *spread_rates*: OPTIONAL: spread_rates is a MxN matrix with spread curve rates defined in columns. Each row contains a specific scenario with different curve structure
- *basis*: OPTIONAL: day-count convention (either basis number between 1 and 11, or specified as string (act/365 etc.)
- *comp_type*: OPTIONAL: compounding type (disc, cont, simple)
- *comp_freq*: OPTIONAL: compounding frequency (1,2,3,4,6,12 payments per year)
- *interp_discount*: OPTIONAL: interpolation method of discount curve (default: linear)
- *interp_spread*: OPTIONAL: interpolatoin method of spread curve (default: linear)
- *npv*: returs a 1xN vector with all net present values per scenario
- *MacDur*: returs a 1xN vector with all Maccaulay durations

See also: timefactor, discount_factor, interpolate_curve.

4.30 rollout_cashflows_oop

`[ret_dates ret_values] = rollout_cashflows_oop (bond_struct)` [Function File] `[ret_dates ret_values] = rollout_cashflows_oop (bond_struct, tmp_nodes, tmp_rates, valuation_date, method_interpolation)` [Function File]

Compute the dates and values of cash flows given definitions for fixed rate bonds, floating rate notes and zero coupon bonds.

In the 'oop' version no input data checks are performed.

Pre-requirements:

- installed octave finance package
- custom functions timefactor, discount_factor, get_forward_rate and interpolate_curve

Input and output variables:

- *bond_struct*: Structure with relevant information for specification of the bond:
  - bond_struct.type: Fixed Rate Bond (FRB), Floating Rate Note (FRN), Zero Coupon Bond (ZCB), Fixed Amortizing Bond (FAB)
  - bond_struct.issue_date: Issue Date of Fixed Rate bond [DD-Mmm-YYYY]
  - bond_struct.maturity_date: Maturity Date of Fixed Rate bond [DD-Mmm-YYYY]
  - bond_struct.compounding_type: Compounding Type [simple,disc,cont]
  - bond_struct.compounding_freq: Compounding Frequency for coupon rates per year [1,2,4,12]
  - bond_struct.term: Term of coupon payments in months [12,6,3,1]
  - bond_struct.day_count_convention: Day Count Convention
  - bond_struct.notional: Notional Amount
  - bond_struct.notional_at_start: Boolean Parameter: Notional Amount paid at start
  - bond_struct.notional_at_end: Boolean Parameter: Notional Amount paid at maturity
  - bond_struct.coupon_rate: Coupon Rate
  - bond_struct.coupon_generation_method: Coupon Generation Method (forward, backward, zero)
  - bond_struct.business_day_rule: Business day rule: days added to payment date
  - bond_struct.business_day_direction: Business day direction: 1
  - bond_struct.enable_business_day_rule: Boolean flag for enabling business day rule
  - bond_struct.spread: Constant Spread applied to forward rates
  - bond_struct.long_first_period: Boolean parameter for long first coupon period, otherwise short (regular) period
  - bond_struct.long_last_period: Boolean parameter for long last coupon period, otherwise short (regular) period

    - bond_struct.last_reset_rate: Last reset rate used for determining fist cash flow of FRN
    - bond_struct.fixed_annuity Boolean parameter: true = annuity loan (total annuity payment is fixed), false = amortizable loan (amortization rate is fixed, interest payments variable)
    - bond_struct.in_arrears Boolean parameter: true = payments at end of period, false: payments at beginning of period (default)
- *tmp_nodes*: only relevant for type FRN: tmp_nodes is a 1xN vector with all timesteps of the given curve
- *tmp_rates*: only relevant for type FRN: tmp_rates is a MxN matrix with curve rates defined in columns. Each row contains a specific scenario with different curve structure
- *ret_dates*: returns a 1xN vector with time in days until all cash flows
- *ret_values*: returs a MxN matrix with all cash flow values at each time step. Each cf row corresponds to rates defined in tmp_rates
- *valuation_date*: Optional: valuation date
- *method_interpolation*: Optional: interpolation method used for retrieving forward rate

See also: timefactor, discount_factor, get_forward_rate, interpolate_curve.

4.31 save_objects

`[`*`riskfactor_struct rf_failed_cell`*`] = save_objects(`*`path_output`*`,`*`riskfactor_struct`*`, `*`instrument_struct`*`,`*`portfolio_struct`*`,`*`stresstest_struct`*`)` [Function File]

Save provided structs for riskfactors, instruments, positions and stresstests.

4.32 scenario_generation_MC

`[`*`R distr_type`*` ] = cenario_generation_MC (`*`corr_matrix`*`, `*`P`*`, `*`mc`*`, `*`copulatype`*`, `*`nu`*`, `*`time_horizon`*`)` [Function File]

Compute correlated random numbers according to Gaussian or Student-t copulas and arbitrary marginal distributions within the Pearson distribution system.

See also: pearsrnd_oct_z, mvnrnd, normcdf, mvtrnd ,tcdf.

4.33 swaption_bachelier

`[`*`SwaptionBachelierValue`*`] = swaption_bachelier (`*`PayerReceiverFlag`*`, `*`F`*`, `*`X`*`, `*`T`*`, `*`r`*`, `*`sigma`*`, `*`m`*`, `*`tau`*`)` [Function File]

Compute the price of european interest rate swaptions according to Bachelier Pricing Functions assuming normal-distributed volatilities. Fast implementation, fully vectorized.

```
C = ((F-X)*N(d1) + sigma*sqrt(T)*n(d1))*exp(-rT) * multiplicator(m,tau)
P = ((X-F)*N(-d1) + sigma*sqrt(T)*n(d1))*exp(-rT) * multiplicator(m,tau)
d1 = (F-X)/(sigma*sqrt(T))
```

Variables:

- *PayerReceiverFlag*: Call / Payer '1' (pay fixed) or Put / Receiver '0' (receive fixed, pay floating) swaption
- *F*: forward rate of underlying interest rate (forward in T years for tau years)
- *X*: strike rate
- *T*: time in days to maturity
- *r*: annual risk-free interest rate (continuously compounded)
- *sigma*: implied volatility of the interest rate measured as annual standard deviation
- *m*: Number of Payments per year (m = 2 -> semi-annual) (continuous compounding is assumed)
- *tau*: Tenor of underlying swap in Years

See also: option_bs.

4.34 swaption_black76

[*SwaptionB76Value*] = swaption_black76 (*PayerReceiverFlag*, *F*, *X*, *T*, *r*, *sigma*, *m*, *tau*) [Function File]

Compute the price of european interest rate swaptions according to Black76 Pricing Functions. Fast implementation, fully vectorized.

```
C = (F*N( d1) - X*N( d2))*exp(-rT) * multiplicator(m,tau)
P = (X*N(-d2) - F*N(-d1))*exp(-rT) * multiplicator(m,tau)
d1 = (log(S/X) + (r + 0.5*sigma^2)*T)/(sigma*sqrt(T))
d2 = d1 - sigma*sqrt(T)
```

Variables:

- *PayerReceiverFlag*: Call / Payer '1' (pay fixed) or Put / Receiver '0' (receive fixed, pay floating) swaption
- *F*: forward rate of underlying interest rate (forward in T years for tau years)
- *X*: strike rate
- *T*: time in days to maturity
- *r*: annual risk-free interest rate (continuously compounded)
- *sigma*: implied volatility of the interest rate measured as annual standard deviation
- *m*: Number of Payments per year (m = 2 -> semi-annual) (continuous compounding is assumed)
- *tau*: Tenor of underlying swap in Years

See also: swaption_bachelier.

4.35 timefactor

`[tf dip dib] = timefactor (d1, d2, basis)` [Function File]

Compute the time factor for a specific time period and day count basis.

Depending on day count basis, the time factor is evaluated as (days in period)/(days in year)

Input and output variables:

- *d1*: number of days until first date (scalar)
- *d2*: number of days until second date (scalar)
- *basis*: day-count basis (scalar)
  - *0* = actual/actual
  - *1* = 30/360 SIA (default)
  - *2* = act/360
  - *3* = act/365
  - *4* = 30/360 PSA
  - *5* = 30/360 ISDA
  - *6* = 30/360 European
  - *7* = act/365 Japanese
  - *8* = act/act ISMA
  - *9* = act/360 ISMA
  - *10* = act/365 ISMA
  - *11* = 30/360E (ISMA)
- *df*: OUTPUT: discount factor (scalar)
- *dip*: OUTPUT: days in period (nominator of time factor) (scalar)
- *dib*: OUTPUT: days in base (denominator of time factor) (scalar)

See also: discount_factor, daysact, yeardays.

Index