



Investment Risk Management (IRM) Component User Manual



Table of Contents

1.	Risk Measures	1
1.1.	VaR	1
1.2.	CVaR	2
1.3.	Calculating 10% best cases.	5
1.4.	Risk at Value	6
2.	Stop-loss (excess) reinsurance	8
3.	Pricing the portfolio with future cash flows	10
3.1.	Creating continuously Rebalanced Portfolio with future cash flows	10
3.2.	Handling costs in portfolio	11
3.3.	VaR of the balanced portfolio with future cash flows	12
4.	Monte Carlo Simulation	14
4.1.	VaR with the Monte Carlo method	14
4.2.	CVaR with the Monte Carlo Method	15
5.	Appendices	16
5.1.	Investment categories (The Netherlands)	16
5.2.	GUISE (The Netherlands) = CLTE	16
5.3.	Risk indicator (The Netherlands)	17
5.4.	IRM Class Diagram	19



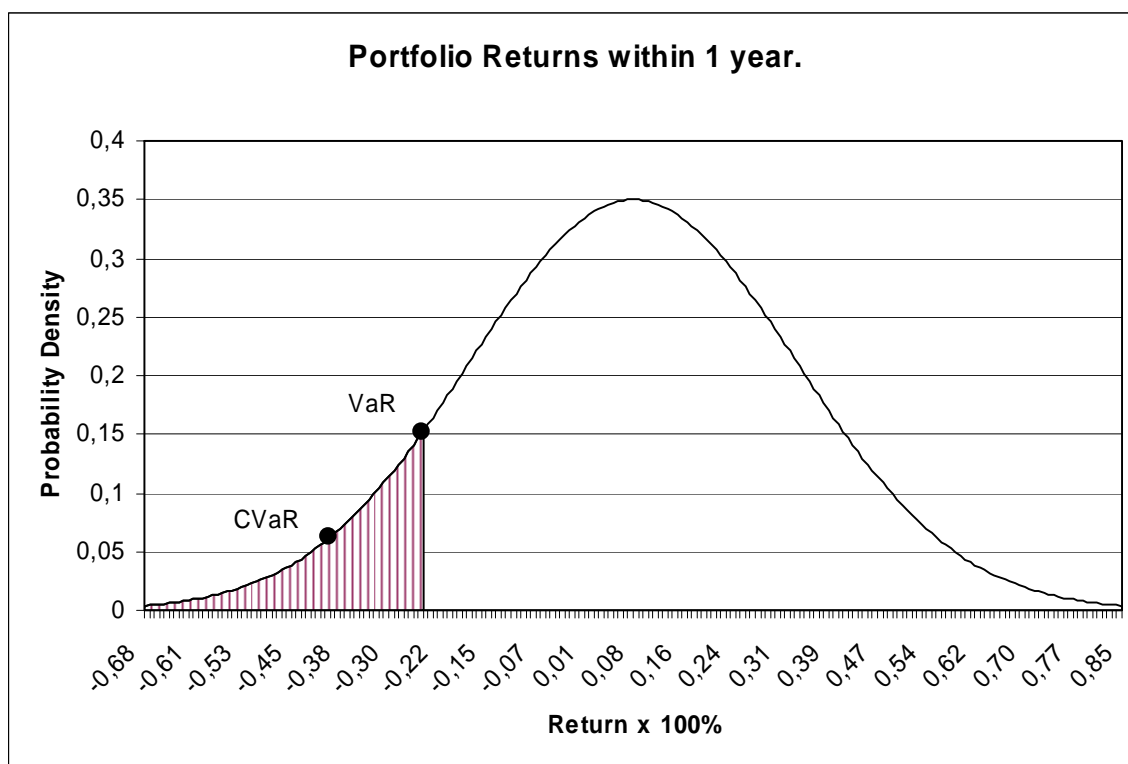
1. Risk Measures

Risk assessment is considered as the initial and periodical step in a risk management process. Risk assessment requires calculations of two components of risk: the magnitude of the potential loss and the probability (p) that the loss will occur.

Risk assessment is possibly the most important step in the risk management process, and may also be the most difficult. A risk with a large potential loss and a low probability of occurring must be treated differently from one with a low potential loss but a high likelihood of occurring.

The IRM component is designed to analyse the risk of a financial portfolio. The most frequently made risk calculations are shown in this manual as practical examples.

Consider a portfolio with an expected interest rate of 0.083 (8.3%). Based on historical observations, average deviations from the expected interest rate are shown in the graph below (volatility=0.255). The hatched area designates 10% of the worst cases. It is clear that in 10% of the worst cases, interest rates can drop to far below zero and form a certain risk.



As with other forms of risk, market risk may be measured in a number of ways. Traditionally, this is done using a Value at Risk (VaR) method.

1.1. VaR

Value at Risk (VaR) is the maximum return or maximum value of the investment in a given worst-case scenario defined as probability.



Although VaR is a very general concept that has broad applications, it is most commonly used by security firms or investment banks to measure the market risk of their asset portfolios (market value at risk). VaR is widely applied in finance for quantitative risk management for many types of risk. VaR only provides information on a maximum value of the portfolio; it does not give any information on how much the actual value can be less than the VaR (severity of loss).

The official definition of VaR gives you an amount of loss in a portfolio. However, we find it more practical to modify it in such a way that it gives you information on the maximum value of the portfolio in worst-case scenarios instead of the minimum loss of the portfolio in worst-case scenarios.

Consider a trading portfolio. Its market value in US dollars today is known to be \$1,000, but its market value tomorrow is unknown. The investment bank holding that portfolio might report that its portfolio value in 5% of the worst-case scenarios had a maximum value of \$996. This implies that under normal trading conditions, the bank can be 95% confident that the value of its portfolio will be greater than \$996 tomorrow (maximum loss \$4). This is the same as saying that there is a 5% risk that the value of its portfolio tomorrow will be less than \$996 (minimum loss \$4).

As we mentioned above, we use VaR as a function of the portfolio value instead of portfolio loss (or profit), so we say that $VaR_{0.05}$ of the portfolio is \$996. In many publications, VaR is specified as the minimum loss in worst-case scenarios, so you may perceive the $VaR_{0.05}$ figure as \$4 instead of \$996. In this manual we use \$996.

The key point to note is that the target risk level (5% in the above example) is the given parameter here; the output from the calculation (\$996 in the example above) is the maximum value at that risk level.

Example:

Product A invests in shares. There are no costs and no guarantees. The initial deposit is €1,000 and the product has a term of 20 years. The $VaR_{0.1}$ for Product A after 20 years is €1,200. This means that after 20 years, in 90% of the cases, an amount larger than €1,200 will be paid out. The amount for the remaining 10% will be less than or equal to €1,200.

This example is calculated with the following code:

```
public void VaR_Example1()
{
    // Define Product A.
    Portfolio portfolio = new Portfolio("Product A", 1000.00, 0.083, 0.255);

    double VaR_Portfolio = IRMMath.VaR(portfolio, 0.10, 20);
    Console.WriteLine("VaR of Product A (20 years) is: {0}", VaR_Portfolio);
    // Output: VaR of Product A (20 years) is €1200.
}
```

1.2. CVaR

Conditional Value at Risk (CVaR) is an alternative to Value at Risk (VaR). It is an average value of the investment in a given worst-case scenario defined as probability. Other terms used for CVaR are Expected Shortfall (ES), Expected Loss



(ELVaR), Expected Tail Loss (ETL), conditional left tail expectation (CLTE), mean excess loss, and worst conditional expectation.

To be consistent in this manual, we use CVaR as an average value of investment instead of average loss of investment when it is used on the portfolio. When it is used on the investment return rate, it gives you an average return rate of the investment for the given worst-case scenario.

Example:

Product A invests in shares. There are no costs and no guarantees. The initial deposit is €1,000 and the product has a term of 5 years. The CVaR for Product A after 5 years is €571. This means that after 5 years, in 10% of the worst cases, an average of €571 of the initial €1,000 will be paid out.

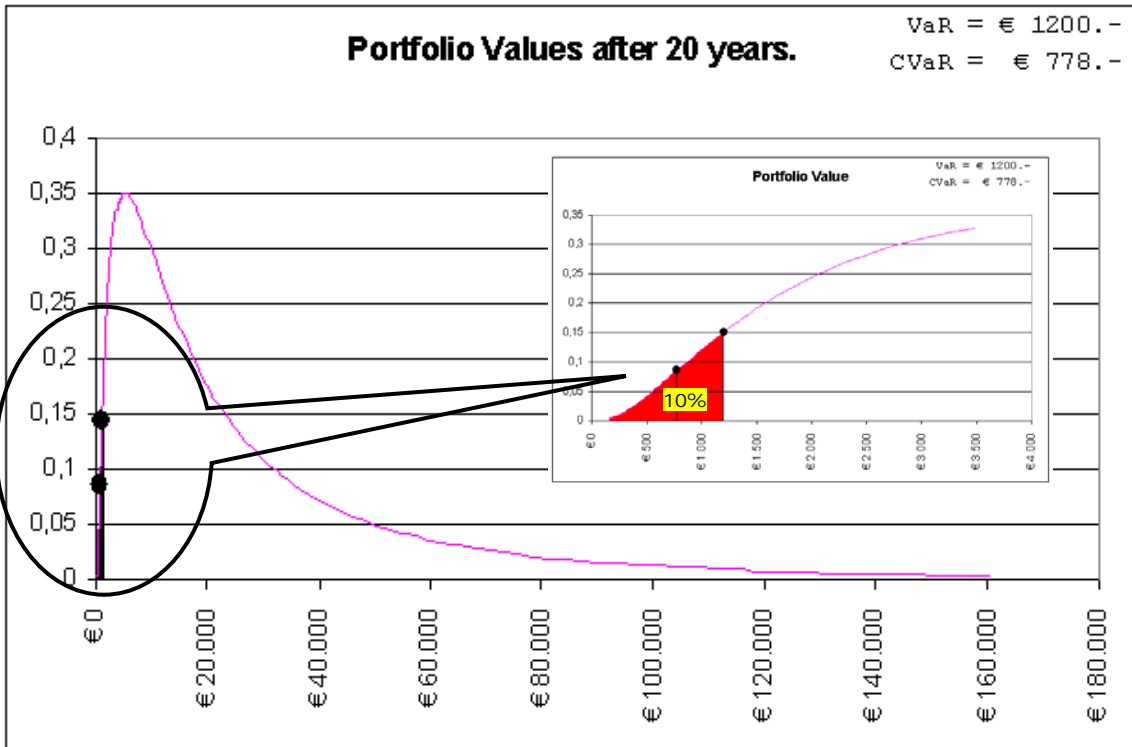
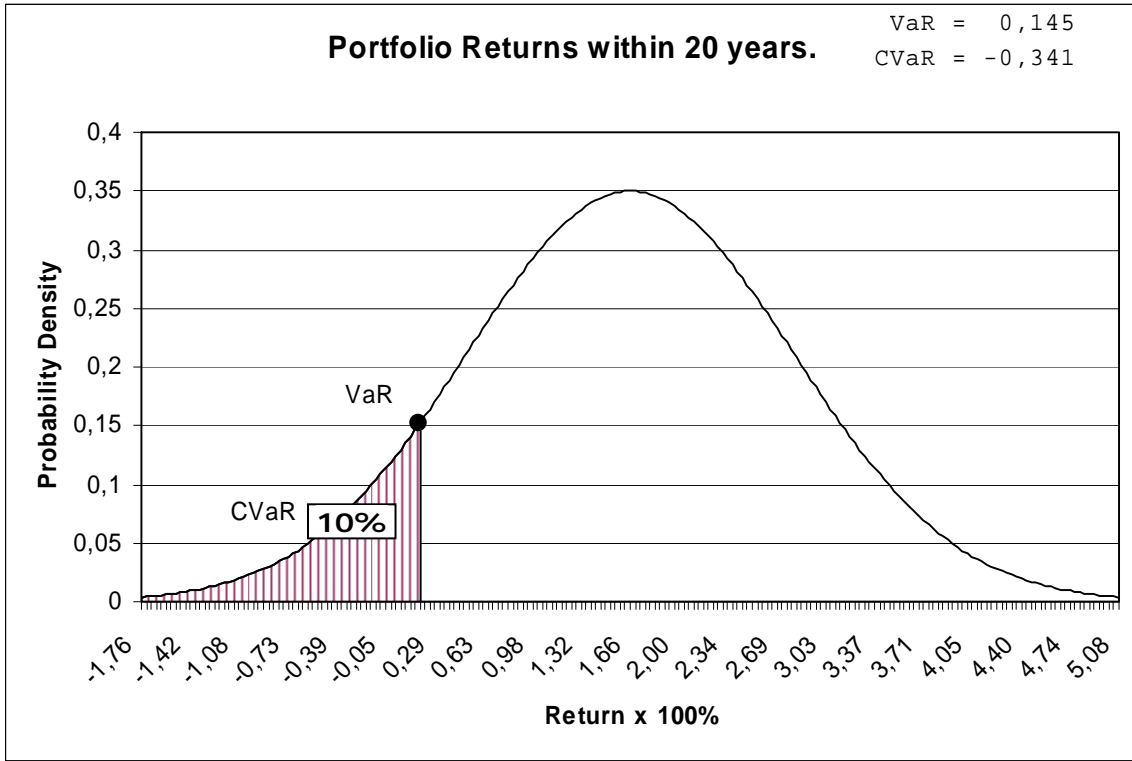
This example is calculated with the following code:

```
public void CVaR_Example1()
{
    // Define Product A.
    Portfolio portfolio = new Portfolio("Product A", 1000.00, 0.083, 0.255);

    // Calculate an average value of the portfolio after 5 years
    // in 10% worst case scenarios.
    double CVaR = IRMMath.CVaR(portfolio, 0.10, 5);
    Console.WriteLine("Product A has a CVaR of {0:C}", CVaR);
    // Output: Product A has a CVaR of €571.34
}
```

CVaR, VaR and GUISE values can be demonstrated with the following example:

Product A invests in shares. There are no costs. The initial deposit is €1,000 and the product has a term of 20 years. After 20 years, the return rate and value of the portfolio have the following probability distribution:

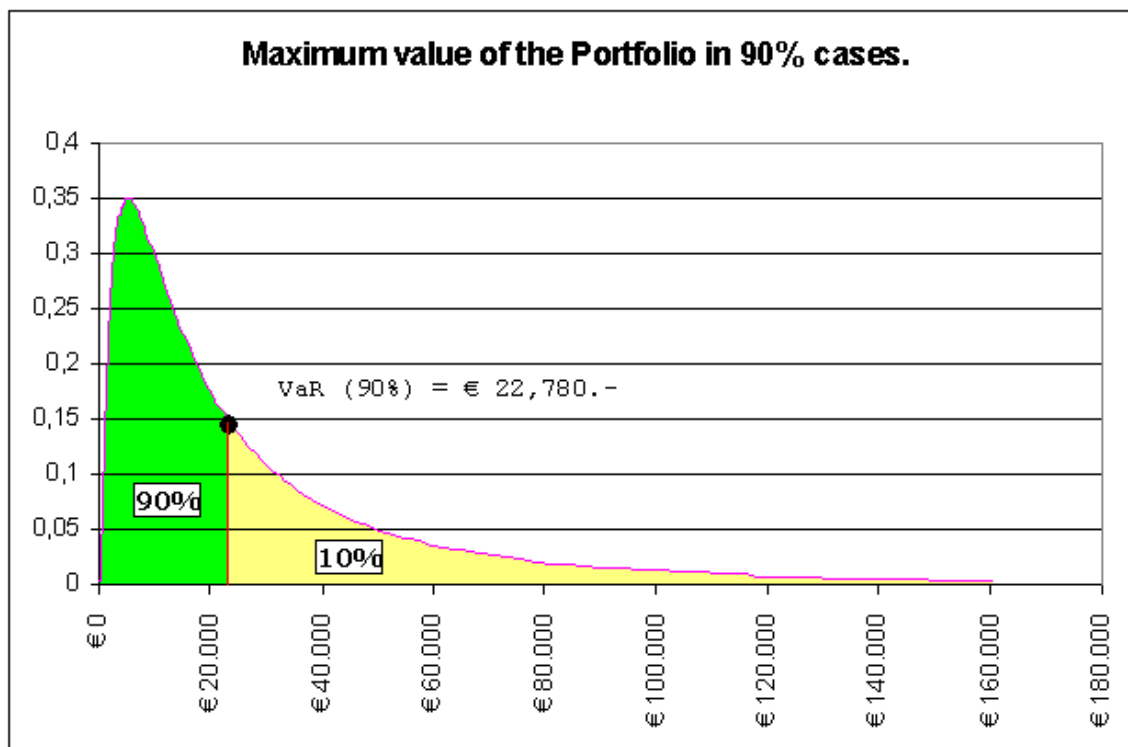




There is a big difference in the shape of the probability distribution between portfolio interest rate and portfolio final value. Indeed, portfolio value is an exponential function of interest rate, so distribution of the portfolio is not normal but log-normal.

1.3. Calculating 10% best cases.

The VaR value can also be used to calculate value for the best scenarios.



The $VaR_{0.9}$ is calculated using the following code:

```
public void VaR_90()  
{  
    // Define Product A.  
    Portfolio portfolio = new Portfolio("Product A", 1000.00, 0.083, 0.255);  
  
    double VaR_Portfolio = IRMMath.VaR(portfolio, 0.90, 20);  
    Console.WriteLine("90 percentile of the Product A (20 years) is:  
        {0:C}", VaR_Portfolio);  
    // Output: 90 percentile of the Product A (20 years) is: €22,679.80  
}
```

The same VaR using the Monte Carlo method can be calculated with the following code:



```
public void VaR_90_MonteCarlo()
{
    // Define Product A.
    Portfolio portfolio = new Portfolio("Product A", 1000.00, 0.083, 0.255);

    // Run Monte Carlo Simulations.
    MonteCarloSimulationCollection simulations =
        MonteCarloSimulation.CreateSimulations(portfolio, 20, 20, 10000);

    // Get statistical output from simulations.
    double[] distribution = simulations.GetDistribution();

    // Analyse VaR.
    double VaR_Portfolio = IRMMath.VaR(distribution, 0.90);

    Console.WriteLine("90 percentile of the Product A (20 years) is: {0:C}",
        VaR_Portfolio);
    // Output: 90 percentile of the Product A (20 years) is: €22,679.80
}
```

1.4. Risk at Value

Risk at Value is the opposite of the Value at Risk method of risk calculation. Here, instead of defining acceptable risk level (as probability), you define a target value of the investment and then analyse the chance (probability) of reaching that value.

In statistics theory, when a distribution of possible values is known, Risk at Value can be calculated using the cumulative distribution function (CDF). The output of the CDF gives you the probability of the value being lower than a specified amount.

By using complimentary CDF ($1.0 - \text{CDF}$), you can calculate the probability of the value being higher than the specified amount.

The Monte Carlo method can be used to generate distribution of values for a given portfolio. To calculate the cumulative distribution function, the `IRMMath.Cdf_Distr()` method is used. To calculate the complimentary distribution function, just subtract the output of `IRMMath.Cdf_Distr()` from 1.0.

Consider the following example.

Product A invests in shares. There are no costs and no guarantees. The initial deposit is €1,000 and the product has a term of 5 years.

- What is the probability (risk) of a portfolio value lower than \$900?
- What is the probability (chance) of a portfolio value higher than \$1,200?

The answer to these questions is calculated with the following code:



```
public void RiskAtValue_MonteCarlo()
{
    // Define Product A.
    Portfolio portfolio = new Portfolio("Product A", 1000.00, 0.083, 0.255);

    // Run Monte Carlo Simulations.
    MonteCarloSimulationCollection simulations =
        MonteCarloSimulation.CreateSimulations(portfolio, 20, 20, 10000);

    // Get statistical output from simulations.
    double[] distribution = simulations.GetDistribution();

    double RaV_Portfolio_900 = IRMMath.Cdf_Distr(distribution, 900);
    Console.WriteLine("Probability of value less than $900 is: {0:0.0%}",
        RaV_Portfolio_900);
    // Output: Probability of value less than $900 is: 6.2%

    double RaV_Portfolio_1200_comp = 1 - IRMMath.Cdf_Distr(distribution, 1200);
    Console.WriteLine("Probability of value greater than $1200 is: {0:0.0%}",
        RaV_Portfolio_1200_comp);
    // Output: Probability of value greater than $1200 is: 90.4%

    Console.WriteLine();
}
```



2. Stop-loss (excess) reinsurance

Stop-loss reinsurance assures its holder that the market value of the portfolio will not be less than a certain amount (guarantee).

The value of the stop-loss reinsurance policy is calculated in the following way:

$R = L \times P$ Where L is an average loss and P is the probability of this loss.

Consider the following trading portfolio:

Product A invests in shares. The initial deposit is \$1,000 and the product has a term of 5 years. There are no costs and there is a guarantee that at least the full initial amount (\$1,000) will be repaid after 5 years. We can estimate the cost of such a guarantee.

We use the Monte Carlo method to analyse the probability of the portfolio value being below \$1,000 (using the Risk at Value method, see §1.4):

```
// Define Product A.
Portfolio portfolio = new Portfolio("Product A", 1000.00, 0.083, 0.255);

// Run Monte Carlo Simulations.
MonteCarloSimulationCollection simulations =
    MonteCarloSimulation.CreateSimulations(portfolio, 5, 5, 10000);

// Get statistical output from simulations.
double[] distribution = simulations.GetDistribution();

// loss probability
double loss_probability = IRMMath.Cdf_Distr(distribution, 1000);
Console.WriteLine("Probability of value less than $1000 is: {0:0%}",
    loss_probability);
// Output: Probability of value less than $1000 is: 23%
```

Now calculate the average value of the portfolio with a calculated risk (23%) using the CVaR method:

```
// average value with calculated probability
double CVaR = IRMMath.CVaR(distribution, loss_probability);
Console.WriteLine("CVaR (below < $1000) of product A is: {0:C}", CVaR);
// Output: CVaR (below < $1000) of product A is: $740
```

So the average loss of this portfolio will be \$1,000 - \$740 = \$260:

```
// average loss with calculated probability
double CVaR_loss = 1000 - CVaR;
Console.WriteLine("Average loss of product A below $1000 is: {0:C}", CVaR_loss);
// Output: Average loss of product A below $1000 is: $260
```

We now know all the parameters we need to calculate the value of the guarantee using the formula above: $R = 260 \times 0.23 = \$60$



```
// Value of the guarantee  
  
double guarantee_value = CVaR_loss * loss_probability;  
Console.WriteLine("Guarantee value is: {0:C}", guarantee_value);  
// Output: Guarantee value is: $60
```

Note: The insurance price is calculated without a safety margin, without insurer costs and is only valid in normal market conditions. It does not include other sources of risk (e.g. disasters).



3. Pricing the portfolio with future cash flows

Pricing the portfolio entails determining its present or future value. When there is only one present cash flow, only the future value of the portfolio is uncertain. The present value is always equal to the initial portfolio value (initial cash flow). The future value is uncertain but the risk is easy to calculate (because it follows well-known log-normal distribution).

With the addition of future cash flows, not only does the present value become uncertain, but the pricing of the portfolio also becomes an important task. Pricing of the portfolio with future cash flows is the subject of many studies and different approximation techniques have been developed.

The straightforward numerical method is by using Monte Carlo simulation (see §4). Despite increasing computational power, the Monte Carlo method remains a very time-consuming calculation.

The other popular methods are well described in the literature as log-normal, reciprocal Gamma, comonotonic upper bound, comonotonic lower bound and 'maximal variance' lower bound approximations.

3.1. Creating continuously Rebalanced Portfolio with future cash flows

A continuously rebalanced portfolio is a portfolio containing a number of assets with given proportions (weights) and these proportions are assumed to remain constant during the life time of the portfolio. In practice, such portfolios should be managed in such a way that all returns and cash flows are divided frequently (e.g. daily) to satisfy the condition of the same proportions between assets.

The risk in a balanced portfolio is easier to calculate without using Monte Carlo simulations. There are a number of approximation methods described in the literature that provide a reasonable level of accuracy.

To create a portfolio with multiple cash flows, we use the following version of Portfolio constructor:

```
public Portfolio(  
    string name,  
    AssetCollection assets,  
    CashFlowCollection transactions,  
    double[,] corrMatrix  
)
```

The first parameter is the name of the portfolio, e.g. the product name displayed in the GUI application. It is optional and can be set to zero.

The second parameter is a collection of assets. Each asset in the collection describes the volatility (risk metric), expected return rate and proportion of asset in the total balanced portfolio.

The third parameter is a collection of cash flows.



The fifth parameter is a correlation matrix, defining linear dependencies between assets in the portfolio. This parameter can be set to zero if assets are uncorrelated (no dependencies).

Example:

Consider the following portfolio containing monthly cash flows (€100) for a term of 5 years (in total $12 * 5 = 60$ cash flows). Of these cash flows, 75% is invested in bonds and 25% in shares (stock).

Such a portfolio can be created with the following code:

```
// Create a collection of cash flows
CashFlowCollection cashFlows = new CashFlowCollection();
for (int i = 0; i < 5 * 12; i++)
{
    double offset = (double)i / 12;
    cashFlows.Add(new CashFlow(100, offset));
}

// Create a collection of assets
AssetCollection assets = new AssetCollection();
assets.Add(new Asset("Bonds", 0.75, 0.042, 0.113));
assets.Add(new Asset("Shares", 0.25, 0.083, 0.255));

// Create a portfolio with assets and cash flows.
Portfolio portfolio = new Portfolio("Product D", assets, cashFlows, null);
```

3.2. Handling costs in portfolio

In the previous example in §3.1, the value of cash flows (€100) and value of return values are net values after subtracting all costs involved in the handling of the portfolio.

When dealing with gross premiums, gross values have to be transformed first prior to calculation.

Fixed annual costs can be subtracted directly from the cash flows at the end (or beginning) of the year.

Fixed monthly costs can be subtracted directly from every monthly cash flow.

Costs defined as a percentage of the cash flows can also be directly subtracted from cash flows.

Annual costs which are relative to the total value of the portfolio can be subtracted from the asset return rates. This subtraction should be in the same proportion as the proportion of assets in the portfolio:

Example:

Consider the following portfolio containing monthly cash flows (€100) for a term of 5 years (in total $12 * 5 = 60$ cash flows). Of these cash flows, 75% is invested in bonds and 25% in shares (stock).

The investor pays €10 per transaction and 1% of the portfolio is an annual fee for managing this portfolio.

€10 fixed costs can be subtracted directly from every cash flow.

1% of the annual fee has to be subtracted from the share interest rate in the following manner:



```
0.083 + 0.25 * Math.Log(0.99)
```

Where 0.083 is the expected return on the shares;

0.25 (25%) is the proportion of the share value to the value of the total portfolio, and 0.99 is 99% of the portfolio value after deducting 1% costs.

The full code for creating such a portfolio is shown here:

```
// Create a collection of cash flows
CashFlowCollection cashFlows = new CashFlowCollection();
for (int i = 0; i < 5 * 12; i++)
{
    double offset = (double)i / 12;
    cashFlows.Add(new CashFlow(100 - 10, offset));
}

// Create a collection of assets
AssetCollection assets = new AssetCollection();
assets.Add(new Asset("Bonds", 0.25, 0.042 + 0.25 * Math.Log(0.99), 0.113));
assets.Add(new Asset("Shares", 0.75, 0.083 + 0.75 * Math.Log(0.99), 0.255));

// Create a portfolio with assets and cash flows.
Portfolio portfolio = new Portfolio("Product D", assets, cashFlows, null);
```

3.3. VaR of the balanced portfolio with future cash flows

Consider the following portfolio containing monthly cash flows (€100) for a term of 5 years (in total $12 * 5 = 60$ cash flows). Of these cash flows 75% is invested in bonds and 25% in shares (stock). What is the $VaR_{0.1}$ of the portfolio value after 5 years?

This example is calculated using the following code:



```
public void VaR_CashFlows()
{
    // Create a collection of cash flows
    CashFlowCollection cashFlows = new CashFlowCollection();
    for (int i = 0; i < 5 * 12; i++)
    {
        double offset = (double)i / 12;
        cashFlows.Add(new CashFlow(100, offset));
    }

    // Create a collection of assets
    AssetCollection assets = new AssetCollection();
    assets.Add(new Asset("Bonds", 0.75, 0.042, 0.113));
    assets.Add(new Asset("Shares", 0.25, 0.083, 0.255));

    // Create a portfolio with assets and cash flows.
    Portfolio portfolio = new Portfolio("Product D", assets, cashFlows, null);

    double portfolioVaR = IRMMath.VaR(portfolio, 0.1, 5);
    Console.WriteLine("VaR (10%) of the Product D after 5 years is: {0:C}",
        portfolioVaR);
    // Output: VaR (10%) of the Product D after 5 years is: €5,523.55
}
```

Using Monte Carlo simulations, the VaR of the cash flows can be calculated with the following code:

```
public void VaR_CashFlows_MonteCarlo()
{
    // Create a collection of cash flows
    CashFlowCollection cashFlows = new CashFlowCollection();
    for (int i = 0; i < 5 * 12; i++)
    {
        double offset = (double)i / 12;
        cashFlows.Add(new CashFlow(100, offset));
    }

    // Create a collection of assets
    AssetCollection assets = new AssetCollection();
    assets.Add(new Asset("Bonds", 0.75, 0.042, 0.113));
    assets.Add(new Asset("Shares", 0.25, 0.083, 0.255));

    // Create a portfolio with assets and cash flows.
    Portfolio portfolio = new Portfolio("Product D", assets, cashFlows, null);

    // Run simulations
    MonteCarloSimulationCollection simulations =
        MonteCarloSimulation.CreateSimulations(portfolio, 5, 60, 10000);

    // Get statistical output
    double[] distribution = simulations.GetDistribution();

    double portfolioVaR = IRMMath.VaR(distribution, 0.1);
    Console.WriteLine("VaR (10%) of the Product D after 5 years is: {0:C}",
        portfolioVaR);
    // Output: VaR (10%) of the Product D after 5 years is: €5,750
}
```

4. Monte Carlo Simulation

In finance, Monte Carlo methods are used to value and analyse basic financial models such as portfolios and investments by simulating the various sources of uncertainty affecting their value, and then determining their average value over the range of resultant outcomes. The advantage of Monte Carlo methods over other techniques increases as the dimensions (sources of uncertainty) of the problem increase.

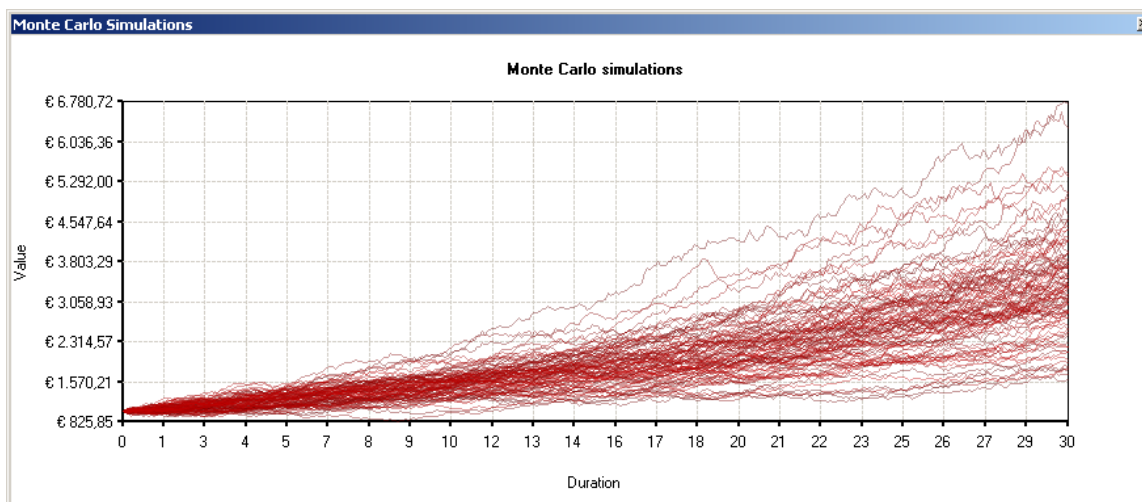


Figure 1; Monte Carlo Simulations

The Monte Carlo method is frequently used to analyse portfolio value. Here, for each simulation, the (correlated) behaviour of the factors underlying the component instruments is simulated over time; the value of the portfolio is calculated and then observed. The various portfolio values are subsequently combined in a histogram (i.e. the portfolio's probability distribution), and the statistical characteristics of the portfolio are then observed. A similar approach is used in calculating value at risk (VaR).

The Monte Carlo method remains the most popular method of solving risk problems. In a significant number of simulations, it produces accurate output and is, therefore, frequently used as a benchmark to compare with other methods.

However, despite the increase of computational power which has been observed in recent years, the computational time remains a serious drawback of Monte Carlo simulations, especially when one has to estimate very high values of quantiles (e.g. the solvency capital of an insurance company can be determined as 99.95% quantile, which is extremely difficult to estimate within a reasonable simulation time).

There are several examples here to demonstrate the use of the Monte Carlo method in finance.

4.1. VaR with the Monte Carlo method

The following example calculates the VaR of the portfolio using the Monte Carlo method.



Product A invests in shares. There are no costs and no guarantees. The initial deposit is €1,000 and the product has a term of 20 years. The VaR for Product A after 20 years is €1,200. This means that after 20 years, in 90% of the cases, an amount larger than €1,200 will be paid out. The amount for the remaining 10% will be less than or equal to €1,200.

This example is calculated with the following code:

```
public void VaR_MonteCarlo()
{
    // Define Product A.
    Portfolio portfolio = new Portfolio("Product A", 1000.00, 0.083, 0.255);

    // Run Monte Carlo Simulations.
    MonteCarloSimulationCollection simulations =
        MonteCarloSimulation.CreateSimulations(portfolio, 20, 20, 10000);

    // Get statistical output from simulations.
    double[] distribution = simulations.GetDistribution();

    // Analyse VaR.
    double VaR_Portfolio = IRMMath.VaR(distribution, 0.10);

    Console.WriteLine("VaR of Product A (20 years) is: {0:C}", VaR_Portfolio);
    // Output: VaR of Product A (20 years) is: €1200.00
}
```

4.2. CVaR with the Monte Carlo Method

The following example calculates the CVaR of the portfolio using the Monte Carlo method.

Product A invests in shares. There are no costs and no guarantees. The initial deposit is €1,000 and the product has a term of 20 years. The CVaR for Product A after 20 years is €778. This means that after 20 years, in 10% of the worst cases, the average value of the portfolio will be €778.

This example is calculated with the following code:

```
public void CVaR_MonteCarlo()
{
    // Define Product A.
    Portfolio portfolio = new Portfolio("Product A", 1000.00, 0.083, 0.255);

    // Run Monte Carlo Simulations.
    MonteCarloSimulationCollection simulations =
        MonteCarloSimulation.CreateSimulations(portfolio, 20, 20, 10000);

    // Get statistical output from simulations.
    double[] distribution = simulations.GetDistribution();

    // Analyse CVaR.
    double CVaR_Portfolio = IRMMath.CVaR(distribution, 0.10);

    Console.WriteLine("CVaR of Product A (20 years) is: {0:C}", CVaR_Portfolio);
    // Output: CVaR of Product A (20 years) is: €778.00
}
```



5. Appendices

5.1. Investment categories (The Netherlands)

Examples in this manual use the following Return Rates and Volatilities¹:

	Return	Volatility	Volatility including currency risk
Deposit	3.7%	0.6%	10.4%
Bonds	4.2%	4.4%	11.3%
Property	6.7%	11.8%	15.7%
Mix fund	6.2%	12.9%	16.6%
Shares	8.3%	25.5%	27.5%
Emerging Markets	8.3%	30.5%	32.2%

5.2. GUISE (The Netherlands) = CLTE

GUISE is a risk measure developed by the Dutch research institution, CentER. The GUISE (*Gemiddelde Uitbetaling In geval van Slechte Eventualiteiten* or 'average payout in the case of unfavourable contingencies' is derived from the CVaR (Conditional Value at Risk).

Another technical name is 'Conditional Left Tail Expectation', or $CLTE_{0.1}$ of the portfolio value in 10% of the worst-case scenarios.

In this manual, GUISE is calculated using the CVaR method with a fixed risk of 10%. It represents the expected payment that a consumer receives if the prices of an investment develop unfavourably. In concrete terms, this means 'What do you get from this product, on average, in 10% of the worst-case scenarios?'

As we mentioned above, this factor is derived from the well-known CVaR function. In fact, CVaR is designed in such a way that, when used on a portfolio, it returns the average value of the portfolio for the worst-case scenario (= CLTE). This value can be used directly as the GUISE factor.

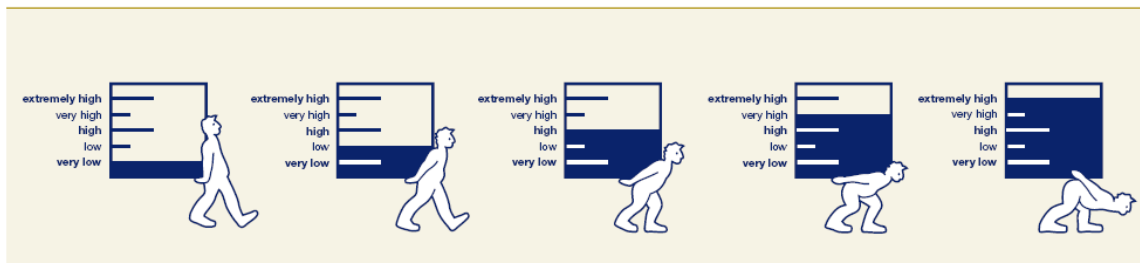
When CVaR is used on an investment return rate, it returns the expected (average) return rate in worst-case scenarios. This is frequently called 'expected loss in case of unfavourable contingencies' or expected shortfall, and measured as a percentage (or perunage).

¹ The parameters were determined by Tilburg University. The method is based on the returns and volatilities provided by Dimson, Marsh & Staunton (Triumph of the Optimists: 101 Years of Global Investment Returns, Princeton University Press) with additional relevant series of indexes. For a detailed description of this method, please refer to Appendix 1.

5.3. Risk indicator (The Netherlands)

The risk indicator, a risk measure developed by CentER, combines GUISE (see §5.2) with the level of guarantee. There are five risk categories, ranging from 'very low' to 'very high'. The risk indicator is represented graphically, showing a figure 'carrying a heavier burden' as the risk becomes higher.

The illustration below shows the graphic risk indicators for the five risk categories. The GUISE is expressed as a percentage of the deposit or the debt. The table below can then be used to look up the category into which a particular product falls.



Risk Indicator	Growth product	Debt product
Very low	Payment of deposit completely guaranteed	Repayment of debt fully guaranteed
Low	Payment of 80% or more of deposit guaranteed; AND GUISE percentage of 95% or greater	Repayment of 80% or more of debt guaranteed; AND GUISE percentage of 90% or greater
High	Less than 80% of deposit guaranteed; AND GUISE percentage of 90% or greater	Less than 80% of debt guaranteed; AND GUISE percentage of 80% or greater
Very high	GUISE percentage between 75% and 90%	GUISE percentage between 65% and 80%
Extremely high	GUISE percentage less than 75%	GUISE percentage less than 65%

Example 1:

Product A invests in shares. There are no costs and no guarantees. The initial deposit is €1,000 and the product has a term of 5 years. The GUISE for Product A after 5 years is €567. This means that after 5 years, in 10% of the worst cases, an average of €567 of the initial €1,000 will be paid out. Product A has a GUISE percentage of $\frac{€571}{€1,000} = 57.1\%$. Since no guarantee is offered, Product A falls into the 'extremely high' risk category.

This example is calculated with the following code:



```
public void RiskIndicator_Example1()
{
    // Define Product A.
    Portfolio portfolio = new Portfolio("Product A", 1000.00, 0.083, 0.255);

    // Calculate GUISE for portfolio with duration 5 years
    // and 10% worst case scenarios.
    double guise = IRMMath.CVaR(portfolio, 0.10, 5);
    double guisePercentage = guise / 1000.00;
    Console.WriteLine("Product A has a GUISE percentage of {0:C} / {1:C}
        = {2:0.0%}", guise, 1000.00, guisePercentage);
    // Output: Product A has a GUISE percentage of €571.34 / €1000.00 = 57.1%

    RiskIndicator risk = IRMMath.GetRiskIndicator(guisePercentage,
        ProductKind.GrowthProduct, 0);
    Console.WriteLine("RiskIndicator for Product A (5 years) is: {0}", risk);
    // Output: RiskIndicator for Product A (5 years) is: ExtremelyHigh
}
```

Example 2:

Product B also invests in shares, has no costs, and has a term of 5 years. The initial deposit is €1,000. The product offers a guarantee for the full deposit (€1,000). Therefore product B falls into the 'very low' risk category.

```
public void RiskIndicator_Example2()
{
    // Define Product B.
    Portfolio portfolio = new Portfolio("Product B", 1000.00, 0.083, 0.255);

    // Calculate GUISE for portfolio with duration 5 years
    // and 10% worst case scenarios.
    double guise = IRMMath.CVaR(portfolio, 0.10, 5);
    double guisePercentage = guise / 1000.00;
    Console.WriteLine("Product B has a GUISE percentage of
        {0:C} / {1:C} = {2:0.0%}", guise, 1000.00, guisePercentage);
    // Output: Product B has a GUISE percentage of €571.34 / €1000.00 = 57.1%

    RiskIndicator risk = IRMMath.GetRiskIndicator(guisePercentage,
        ProductKind.GrowthProduct, 100);
    Console.WriteLine("RiskIndicator for Product B (5 years) is: {0}", risk);
    // Output: RiskIndicator for Product B (5 years) is: VeryLow
}
```



5.4. IRM Class Diagram

