Package ‘sstModel’

May 4, 2018

Title Swiss Solvency Test (SST) Standard Models

License GPL-3 + file LICENSE

Version 1.0.0

Description Framework for the implementation of solvency related computations based on standard models for the Swiss Solvency Test (SST), a risk-based capital standard for Swiss insurance companies. Allows Monte Carlo simulation of market risk, some insurance risks and their aggregation. Additional toolbox for preprocessing computations. Convenient 'shiny' GUI combined with a parser for an input 'excel' (.xlsx) template to simplify model configuration, data fill-in and results visualization.

Depends R (>= 3.3.0)

Imports data.table (>= 1.10.4-3), stats, utils, tools, readxl (>= 1.0.0), openxlsx (>= 4.0.17), MASS, shiny (>= 1.0.5), shinydashboard (>= 0.6.1)

Encoding UTF-8

LazyData true

Suggests testthat, knitr, covr

RoxygenNote 6.0.1

VignetteBuilder knitr

NeedsCompilation no

Author Loris Michel [aut],
Melvin Kianmanesh Rad [aut],
Adrien Lamit [aut],
Michael Schmutz [cre],
Swiss Financial Market Supervisory Authority FINMA [cph]

Maintainer Michael Schmutz <michael.schmutz@finma.ch>

Repository CRAN

Date/Publication 2018-05-03 22:21:08 UTC
R topics documented:

  sstModel-package .............................................. 8
  aggregateRisks .................................................. 9
  asset ........................................................... 9
  assetForward ................................................... 10
  cashflow ........................................................ 12
  changeBaseCurrency ............................................ 13
  check ........................................................... 14
  check.asset ..................................................... 14
  check.assetForward ........................................... 15
  check.cashflow .................................................. 16
  check.delta ..................................................... 16
  check.fxForward ............................................... 17
  check.health ................................................... 18
  check.liability ............................................... 18
  check.life ..................................................... 19
  check.macroEconomicScenarios ............................... 20
  check.nonLifeRisk ............................................. 20
  check.participation ............................................ 21
  check.scenarioRisk ............................................ 22
  check.standalone ............................................... 22
  compute ........................................................ 23
  compute.healthRisk ............................................ 24
  compute.lifeRisk ............................................... 24
  compute.macroEconomicScenarios ............................. 25
  compute.marketRisk .......................................... 26
  compute.nonLifeRisk .......................................... 26
  compute.participationRisk .................................... 27
  compute.scenarioRisk .......................................... 28
  compute.sstModel ............................................... 29
  computeConstant .............................................. 29
  conditionalReordering ........................................ 30
  containsHealth ................................................ 31
  containsHealth.sstOutput ..................................... 32
  containsInsurance .............................................. 32
  containsInsurance.sstOutput .................................. 33
  containsLife ................................................... 33
  containsLife.sstOutput ........................................ 34
  containsMarket ................................................ 35
  containsMarket.sstOutput ..................................... 35
  containsNonLife ............................................... 36
  containsNonLife.sstOutput ..................................... 36
  containsParticipation ......................................... 37
  containsParticipation.sstOutput ............................... 37
  containsScenario .............................................. 38
  containsScenario.sstOutput ..................................... 39
  creditRisk ...................................................... 39
R topics documented:

creditRisk.sstOutput .......................................................... 40
currency ................................................................. 40
currencyIsIn ............................................................. 41
currencyIsIn.standalone .................................................. 42
delta ................................................................. 42
equity ................................................................. 43
equityIsIn ............................................................. 45
equityIsIn.standalone .................................................. 45
excelToSstModel .......................................................... 46
expectedShortfall .......................................................... 46
format.asset .............................................................. 47
format.assetForward ...................................................... 47
format.cashflow ........................................................... 48
format.delta .............................................................. 49
format.fxForward .......................................................... 49
format.health .............................................................. 50
format.healthRisk .......................................................... 50
format.liability ............................................................ 51
format.life ............................................................... 51
format.lifeRisk ............................................................. 52
format.marketRisk .......................................................... 53
format.nonLifeRisk .......................................................... 53
format.participation ......................................................... 54
format.participationRisk .................................................... 54
format.portfolio ............................................................ 55
format.scenarioRisk .......................................................... 55
format.sstModel ............................................................ 56
format.sstOutput ........................................................... 57
format.standalone ........................................................... 57
format.summary.portfolio ..................................................... 58
format.summary.sstModel .................................................. 58
format.summary.sstOutput .................................................. 59
fxForward ................................................................. 59
generateError ............................................................... 61
generateExpression .......................................................... 61
generateExpression.portfolio .............................................. 62
generateFunction ........................................................... 63
generateFunction.portfolio .................................................. 63
getCurrencyId .............................................................. 64
getCurrencyId.marketRisk ..................................................... 65
getCurrencyName ............................................................ 65
getCurrencyName.marketRisk .................................................. 66
getCurrencyScale ............................................................. 67
getCurrencyScale.marketRisk .................................................. 67
getDeltaId ................................................................. 68
getDeltaId.marketRisk ....................................................... 68
getDrbc ................................................................. 69
getDrbc.sstOutput ........................................................... 69
getEquityId  ................................................................. 70
getEquityId.marketRisk .................................................. 71
getEquityName ............................................................... 71
getEquityName.marketRisk ................................................. 72
getEquityScale .............................................................. 73
getEquityScale.marketRisk ............................................... 73
getHealthId ................................................................. 74
getHealthQuantile .......................................................... 74
getHealthRisk .............................................................. 75
getHealthRisk.sstOutput ................................................ 75
getInitialFX ................................................................. 76
getInitialFX.marketRisk .................................................... 76
getInitialRate ............................................................... 77
getInitialRate.marketRisk ................................................ 77
getInitialSpread ............................................................ 78
getInsuranceRisk ........................................................... 78
getInsuranceRisk.sstOutput .............................................. 79
getLifeId ..................................................................... 79
getLifeId.lifeRisk .......................................................... 80
getLifeQuantile ............................................................. 80
getLifeQuantile.lifeRisk .................................................. 81
getLifeRisk ................................................................. 81
getLifeRisk.sstOutput ..................................................... 82
getMappingTime ............................................................ 82
getMappingTime.marketRisk .............................................. 83
getMarketParticipationRisk ............................................. 83
getMarketParticipationRisk.sstOutput .................................. 84
getMarketRisk .............................................................. 85
getMarketRisk.sstOutput ................................................ 85
getNonLifeRisk ............................................................. 86
getNonLifeRisk.sstOutput ................................................ 86
getParticipation ............................................................ 87
getParticipation.sstOutput ............................................... 87
getRateId .................................................................... 87
getRateId.marketRisk ..................................................... 88
getRateName ............................................................... 88
getRateName.marketRisk .................................................. 89
getRateScale ............................................................... 89
getRateScale.marketRisk ................................................ 90
getScenarioRisk ........................................................... 90
getScenarioRisk.sstOutput .............................................. 91
getSpreadId .............................................................. 92
getSpreadId.marketRisk .................................................. 92
getSpreadName ........................................................... 93
getSpreadName.marketRisk ............................................... 94
getSpreadScale ........................................................... 95
getSpreadScale.marketRisk .............................................. 95
health ................................................................. 96
healthRisk .......................................................... 97
initialFX .......................................................... 98
initialRate ......................................................... 98
initialSpread ...................................................... 99
intToGroups ....................................................... 99
is.asset ............................................................. 100
is.assetForward ................................................ 100
is.cashflow ......................................................... 101
is.currency ........................................................ 101
is.delta ............................................................. 102
is.equity ........................................................... 102
is.fxForward ....................................................... 103
is.health ........................................................... 103
is.healthRisk ....................................................... 104
is.insuranceItem ................................................ 104
is.insuranceRisk ................................................ 105
is.item .............................................................. 105
is.liability ........................................................ 106
is.life .............................................................. 106
is.lifeRisk ........................................................ 107
is.macroEconomicScenarios ................................. 107
is.mappingTable ................................................. 108
is.marketItem ..................................................... 108
is.marketRisk ..................................................... 109
is.nonLifeRisk .................................................... 109
is.participation .................................................. 110
is.participationRisk ............................................ 110
is.pcRate .......................................................... 111
is.portfolio ........................................................ 111
is.rate .............................................................. 112
is.risk .............................................................. 112
is.riskFactor ...................................................... 113
is.scenarioRisk .................................................. 113
is.spread .......................................................... 114
is.sstModel ......................................................... 114
is.sstOutput ....................................................... 115
is.standalone ..................................................... 115
itemListToExpression .......................................... 116
itemListToFunction ............................................ 116
keywordToTable ................................................ 117
keywordToTransposedTable ................................. 118
keywordToValue ................................................ 118
launchDashboard ................................................ 119
liability ............................................................ 120
life ................................................................. 121
lifeRisk ............................................................ 122
logNormalExpression ......................................... 123
macroEconomicScenarios ................................. 123
mappingTable . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 124
mappingTime . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 125
marketRisk . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 125
marketValueMargin . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 126
marketValueMargin.sstOutput . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 127
mvnLife . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 127
na.rm . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 128
newtonRaphson . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 128
nonLifeRisk . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 129
participation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 131
participationRisk . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 132
pcRate . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 132
portfolio . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 133
print.asset . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 135
print.assetForward . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 136
print.cashflow . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 137
print.delta . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 137
print.fxForward . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 138
print.health . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 139
print.healthRisk . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 140
print.liability . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 141
print.life . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 141
print.lifeRisk . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 142
print.marketRisk . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 143
print.nonLifeRisk . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 144
print.participation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 145
print.participationRisk . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 145
print.portfolio . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 146
print.scenarioRisk . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 147
print.sstModel . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 148
print.sstOutput . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 149
print.standalone . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 149
print.summary.portfolio . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 150
print.summary.sstModel . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 150
print.summary.sstOutput . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 151
rate . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 151
rateIsIn . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 153
rateIsIn.standalone . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 153
removePerfectCorr . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 154
riskCapital . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 155
riskCapital.sstOutput . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 155
riskFactorToExpression . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 156
scenarioRisk . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 156
simulate.healthRisk . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 157
simulate.lifeRisk . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 158
simulate.marketRisk . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 158
simulate.nonLifeRisk . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 159
simulate.participationRisk . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 160
```r
R topics documented:
simulate.scenarioRisk .......................... 160
splitComma ........................................ 161
spread ........................................... 162
spreadIsIn ....................................... 163
spreadIsIn.standalone ......................... 163
sstModel .......................................... 164
sstModel_check ................................. 165
sstModel_news .................................... 166
sstRatio ........................................... 166
sstRatio.sstOutput ............................... 167
standalone ....................................... 167
standaloneExpectedShortfall ................. 169
standaloneExpectedShortfall.sstOutput ..... 169
summary.asset ................................. 170
summary.assetForward ......................... 171
summary.cashflow .............................. 171
summary.delta ................................... 172
summary.fxForward ............................. 173
summary.health ................................. 174
summary.healthRisk ............................ 174
summary.liability ............................. 175
summary.life ..................................... 176
summary.lifeRisk ............................... 177
summary.marketRisk ............................ 178
summary.nonLifeRisk ........................... 178
summary.participation ......................... 179
summary.participationRisk .................... 180
summary.portfolio ............................ 181
summary.scenarioRisk ......................... 181
summary.sstModel ............................. 182
summary.sstOutput ............................. 183
summary.standalone .................. ........... 184
tableToAssetForward ......................... 185
tableToAssets ................................... 185
tableToCashflow .............................. 186
tableToFxForward ............................. 186
tableToLiability .............................. 187
targetCapital ................................. 187
targetCapital.sstOutput ....................... 188
translate ........................................ 188
translate.sstOutput ........................... 189
valExpression ................................. 189
valExpression.asset ......................... 190
valExpression.assetForward ................. 190
valExpression.cashflow ....................... 191
valExpression.delta ......................... 192
valExpression.fxForeward .................... 192
valExpression.health ......................... 193
```
valExpression.liability .................................................. 194
valExpression.life ......................................................... 194
valFunction ............................................................... 195
valFunction.asset ......................................................... 196
valFunction.assetForward ............................................... 197
valFunction.cashflow .................................................... 197
valFunction.delta ........................................................ 198
valFunction.fxForward .................................................. 199
valFunction.liability ..................................................... 199
valInfo ................................................................. 200
valInfo.asset ............................................................ 201
valInfo.assetForward ................................................... 202
valInfo.cashflow ........................................................ 202
valInfo.delta ............................................................. 203
valInfo.fxForward ......................................................... 204
valInfo.health ............................................................ 205
valInfo.liability .......................................................... 205
valInfo.life ............................................................... 206
valueAtRisk .............................................................. 206
volaToExpectedShortfall ................................................ 207
write.sstOutput ......................................................... 208

Index  ................................................................. 209

sstModel-package  

Implementation of the Swiss Solvency Test (SST) Standard Models.

Description

Framework for the implementation of solvency related computations based on standard models for
the Swiss Solvency Test (SST), a risk-based capital standard for Swiss insurance companies. Allows
Monte Carlo simulation of market risk, some insurance risks and their aggregation. Additional
toolbox for preprocessing computations. Convenient shiny GUI combined with a parser for an
input excel (.xlsx) template to simplify model configuration,

Main Functionality the R-package

The main functionality of the R-package is the construction of an sstModel object, i.e. an instance
of the Swiss Solvency Test (SST) standard model (all parameters needed to create such an instance
can be understood with their respective help pages). We can then simulate from the model with the
method compute to obtain an sstOutput instance. Solvency figures can finally be computed on
this last instance (like riskCapital, targetCapital, marketValueMargin, and sstRatio).

See Also

sstModel
aggregateRisks

Risk Aggregation Helper

Description

This function aggregates market, life, health and nonLife insurance risks using a simple or conditional reordering scheme based on Gaussian copulas.

Usage

aggregateRisks(risks, model)

Arguments

risks  data.table object.
model  sstModel S3 object.

Value

None (used for side-effects).

asset

Constructing an Asset with Direct Market Price

Description

Constructor for the S3 class asset. It allows to build for an asset position with direct market price known under the name "Aktiven mit direkt marktabhängigen Preisen" in the FINMA technical document "SST-Marktrisiko und -Aggregation Technische Beschreibung".

Usage

asset(type, currency, value)

Arguments

type  character value of length one representing the type of the asset position. This parameter relates to the "Preisrisikofaktor" index i in the FINMA document "SST-Marktrisiko und -Aggregation Technische Beschreibung". type cannot be one of the following reserved character:

- "currency"
- "rate"
- "pcRate"
- "spread"
currency

Character value of length one representing the currency in which the asset is valued. This parameter relates to the "Fremdwährungsrisikofaktor" index \( j \) in the FINMA document "SST-Marktrisiko und -Aggregation Technische Beschreibung".

value

Non-zero numeric value of length one representing the exposure in the underlying asset. This must be expressed in the same currency as currency. Note that if value is negative the position is interpreted as a short position. If the value is set to 0, a warning will be triggered. This parameter corresponds to the quantity \( \hat{E}_{0,i,j} \) for asset with direct market price in the FINMA document "SST-Marktrisiko und -Aggregation Technische Beschreibung".

Value

An S3 object, instance of the class asset.

See Also

summary.asset, print.asset.

Examples

```r
# Creating new assets.
asset1 <- asset("equity", "CHF", 1000)
asset2 <- asset("hedge fund", "EUR", 2000)
```

assetForward

Constructing an Index-Forward

Description

Constructor for the S3 class assetForward. It allows to build for an index-forward referred under the name "Index-Forward" in the FINMA technical document "SST-Marktrisiko und -Aggregation Technische Beschreibung".

Usage

```r
assetForward(type, currency, time, exposure, price, position)
```
**Arguments**

- **type** character value of length one representing the type of the underlying asset position. This parameter relates to the index $i$ in the valuation formula of index-forwards in the FINMA document "SST-Marktrisiko und -Aggregation Technische Beschreibung". This parameter is the same as the "Preisrisikofaktor" index 1 for asset valuation in the same document. type cannot be one of the following reserved character:
  - "currency"
  - "rate"
  - "pcRate"
  - "spread"

- **currency** character value of length one representing the currency in which the underlying asset is valuated. This parameter relates to the "Fremdwährungsrisikofaktor" index $j$ in the FINMA document "SST-Marktrisiko und -Aggregation Technische Beschreibung".

- **time** strictly positive integer value of length one representing the time-to-maturity from $t = 0$. This parameter relates to the variable $\tau$ in valuation formula for assetForwards in the FINMA document "SST-Marktrisiko und -Aggregation Technische Beschreibung".

- **exposure** strictly non-zero numeric value of length one. The exposure in the underlying asset covered by the forward contract, this must be expressed in the same currency as currency. This parameter corresponds to the quantity $E_{0,i,j}$ for assetForwards in the FINMA document "SST-Marktrisiko und -Aggregation Technische Beschreibung". If exposure is set to 0, a warning will be triggered.

- **price** numeric value of length one representing the forward price. This parameter relates to the assetForward variable

  $$\hat{F}^j_{\tau}$$

  in the FINMA document "SST-Marktrisiko und -Aggregation Technische Beschreibung". This must be expressed in the same currency as currency.

- **position** character value of length one. This can be either "long" or "short" according to the definition of long and short forwards in the FINMA document "SST-Marktrisiko und -Aggregation Technische Beschreibung".

**Value**

an S3 object, instance of the class fxForward.

**Note**

The underlying equity shall be defined using asset.
See Also

summary.assetForward, print.assetForward.

Examples

# Creating new assetForwards.
asset.forward.1 <- assetForward("equity", "EUR", 1, 1000, 1200, "long")
asset.forward.2 <- assetForward("private real estate", "CHF", 7, 100, 90,
  "short")

Description

Constructor for the S3 class cashflow. It allows to build for a fixed-income-asset referred under the name "Fixed-Income-Assets" in the FINMA technical document "SST-Marktrisiko und -Aggregation Technische Beschreibung".

Usage

cashflow(time, currency, rating, spread, value)

Arguments

time stricly positive integer value of length one representing the time-to-maturity.
This parameter relates to the "Restlaufzeit" cashflow variable tau in the FINMA
document "SST-Marktrisiko und -Aggregation Technische Beschreibung".

currency character value of length one representing the currency in which the fixed-
income-asset is labeled. This parameter relates to the "Fremdwährungsrisikofaktor" cashflow index j in the FINMA document "SST-Marktrisiko und -Aggregation Technische Beschreibung".

rating character value of length one representing the rating associated to the fixed-
income-asset. This parameter relates to the "Rating" cashflow variable r in the FINMA document "SST-Marktrisiko und -Aggregation Technische Beschreibung".

spread a numeric value of length one representing the initial spread corresponding to the
fixed-income-asset. This parameter relates to the cashflow variable $S(0, j, r)$ in the
FINMA document "SST-Marktrisiko und -Aggregation Technische Beschreibung". A warning is triggered if spread is below -0.1 or above 0.3.

value non-zero numeric value of length one representing the expected cashflow at time
time for a fixed-income-asset with rating rating. This must be expressed in the
same currency as currency. If value is negative, then the cashflow is interpreted as a liability. This parameter corresponds to the cashflow quantity

$CF^A_{r,j}$
Value

an S3 object, instance of the class cashflow.

See Also

summary.cashflow, print.cashflow.

Examples

# Creating new cashflows.
cashflow1 <- cashflow(1L, "USD", "AAA", 0.1, 1000)
cashflow2 <- cashflow(2L, "EUR", "BB", 0.1, 2000)

changeBaseCurrency

Change Covariance Matrix According to Change of Base Currency

Description

This function allow to change the base risk factor covariance matrix according to a change of base currency, the function also update the mapping.table and ask the user to provide new names for the new fx base risks.

Usage

changeBaseCurrency(cov.mat, mapping.table, target.currency, mapping.name)

Arguments

cov.mat     matrix value corresponding to the covariance matrix of base risk factors. This matrix should have an attribute named "base.currency" indicating the actual base currency in which the covariance matrix is expressed.
mapping.table S3 object of class mappingTable that should be coherent with the cov.mat.
target.currency character value of length one indicating the new base currency, this should exists in the mapping.table.
mapping.name data.frame indicating the mapping towards new name in the covariance matrix and in the mapping.table for the new fx rate with two columns:
  • old.name: the names of the old risk factors in the covariance matrix.
  • new.name: the new names of these risk factors.
check.asset

**Value**

a list with two named fields:

- `cov.mat`: the new covariance matrix.
- `mapping.table`: the new mapping.table.

---

**Description**

check is a generic S3 method for S3 classes inheriting from item. It is a logical method checking if the item is well-defined with respect to a risk (i.e. that all information necessary for valuating the item is available).

**Usage**

check(object, ...)

**Arguments**

- `object` an S3 object to check.
- `...` additional parameters.

**Value**

a logical value.

---

check.asset

**Description**

check is a generic S3 method for S3 classes inheriting from item. It is a logical method checking if the item is well-defined with respect to a risk (i.e. that all information necessary for valuating the item is available).

**Usage**

```r
## S3 method for class 'asset'
check(object, market.risk, ...)
```
check.assetForward

Arguments

object S3 object of class asset.
market.risk S3 object of class marketRisk created using the constructor marketRisk.
... additional arguments.

Value

a logical value, is the asset consistent with the marketRisk?

See Also

check, asset, marketRisk.

check.assetForward Checking Consistency of an Index-Forward with a MarketRisk

Description

check is a generic S3 method for S3 classes inheriting from item. It is a logical method checking if the item is well defined with respect to a risk (i.e. that all information necessary for valuating the item is available).

Usage

## S3 method for class 'assetForward'
check(object, market.risk, ...)

Arguments

object S3 object of class assetForward.
market.risk S3 object of class marketRisk created using the constructor marketRisk.
... additional arguments.

Value

a logical value, is the asset forward consistent with the marketRisk?
check.cashflow  Checking Consistency of a Fixed-Income-Asset with a MarketRisk

Description
check is a generic S3 method for S3 classes inheriting from item. It is a logical method checking if the item is well defined with respect to a risk (i.e. that all information necessary for valuating the item is available).

Usage
## S3 method for class 'cashflow'
check(object, market риск, ...)

Arguments
- object: S3 object of class cashflow.
- market риск: S3 object of class market риск created using the constructor market риск.
- ...: additional arguments.

Value
a logical value, is the cashflow consistent with the market риск?

See Also
check, cashflow, market риск

check.delta  Checking Consistency of a Delta-Normal Remainder Term with a MarketRisk

Description
check is a generic S3 method for S3 classes inheriting from item. It is a logical method checking if the item is well defined with respect to a risk (i.e. that all information necessary for valuating the item is available).

Usage
## S3 method for class 'delta'
check(object, market риск, ...)

Arguments
- object: S3 object of class delta.
- market риск: S3 object of class market риск created using the constructor market риск.
- ...: additional arguments.
Arguments

- **object**: S3 object of class `delta`.
- **market.risk**: S3 object of class `marketRisk` created using the constructor `marketRisk`.
- **...**: additional arguments.

Value

- a logical value, is the `delta` consistent with the `marketRisk`?

See Also

- `check`, `delta`, `marketRisk`.

---

**check.fxForward**

Checking Consistency of a FX-Forward with a MarketRisk

Description

`check` is a generic S3 method for S3 classes inheriting from `item`. It is a logical method checking if the item is well defined with respect to a risk (i.e. that all information necessary for valuating the item is available).

Usage

```r
## S3 method for class 'fxForward'
check(object, market.risk, ...)
```

Arguments

- **object**: S3 object of class `fxForward`.
- **market.risk**: S3 object of class `marketRisk` created using the constructor `marketRisk`.
- **...**: additional arguments.

Value

- a logical value, is the `fxForward` consistent with the `marketRisk`?
Description
check is a generic S3 method for classes inheriting from item. It is a logical method checking if the item is well defined with respect to a risk (i.e. that all information necessary for valuating the item is available).

Usage
## S3 method for class 'health'
check(object, market.risk, health.risk, ...)

Arguments
- object: S3 object of class health.
- market.risk: S3 object of class marketRisk created using the constructor marketRisk.
- health.risk: S3 object of class healthRisk, created using the constructor healthRisk.
- ...: additional arguments.

Value
a logical value, is the health item consistent with the marketRisk and the healthRisk?

See Also
check, health.

Description
check is a generic S3 method for S3 classes inheriting from item. It is a logical method checking if the item is well defined with respect to a risk (i.e. that all information necessary for valuating the item is available).

Usage
## S3 method for class 'liability'
check(object, market.risk, ...)

Arguments
- object: S3 object of class liability.
check.life

Arguments

- **object**: S3 object of class liability.
- **marketRisk**: S3 object of class marketRisk created using the constructor `marketRisk`.
- ...
  - additional arguments.

Value

a logical value, is the liability consistent with the marketRisk?

See Also

- `check`, `liability`, `marketRisk`.

check.life  
*Checking Consistency of a Life Delta-Normal Remainder Term with a MarketRisk and a HealthRisk*

Description

check is a generic S3 method for classes inheriting from item. It is a logical method checking if the item is well defined with respect to a risk (i.e. that all information necessary for valuating the item is available).

Usage

```r
## S3 method for class 'life'
check(object, marketRisk, lifeRisk, ...)
```

Arguments

- **object**: S3 object of class life.
- **marketRisk**: S3 object of class marketRisk created using the constructor `marketRisk`.
- **lifeRisk**: S3 object of class lifeRisk, created using the constructor `lifeRisk`.
- ...
  - additional arguments.

Value

a logical value, is the life item consistent with the marketRisk and the healthRisk?

See Also

- `check`, `life`.
check.macroEconomicScenarios

Checking Macro Economic Scenarios

Description

Checking Macro Economic Scenarios

Usage

## S3 method for class 'macroEconomicScenarios'
check(object, market.risk, portfolio, ...)

Arguments

object an S3 object of class macroEconomicScenario.
market.risk an S3 object of class marketRisk.
portfolio an S3 object of class portfolio.
... additional arguments.

Value

a logical value.

check.nonLifeRisk

Checking Consistency of a nonLifeRisk with a MarketRisk

Description

check is a generic S3 method for classes inheriting from item as well as nonLifeRisk. It is a logical method checking if the item is well defined with respect to a risk (i.e. that all information necessary for valuating the item/scenario is available).

Usage

## S3 method for class 'nonLifeRisk'
check(object, market.risk, ...)

Arguments

object S3 object of class nonLifeRisk.
market.risk S3 object of class marketRisk created using the constructor marketRisk.
... additional parameters.
check.participation

Value

   a logical value, is the nonLifeRisk consistent with the marketRisk?

See Also

   check, nonLifeRisk.

---

check.participation  Checking Consistency of a Participation with a MarketRisk

Description

   check is a generic S3 method for classes inheriting from item. It is a logical method checking if the
   item is well defined with respect to a risk (i.e. that all information necessary for valuating the item
   is available).

Usage

   ## S3 method for class 'participation'
   check(object, marketRisk, ...)

Arguments

   object       S3 object of class participation.
   marketRisk   S3 object of class marketRisk created using the constructor marketRisk.
   ...          additional arguments.

Value

   a logical value, is the participation consistent with the marketRisk?

See Also

   check, participation.
check.scenarioRisk  

Checking Consistency of a ScenarioRisk with a MarketRisk

Description
check is a generic S3 method for classes inheriting from item. It is a logical method checking if the item is well defined with respect to a risk (i.e., that all information necessary for valuating the item is available).

Usage
```r
## S3 method for class 'scenarioRisk'
check(object, marketRisk, ...)
```

Arguments
- `object` S3 object of class scenarioRisk.
- `marketRisk` S3 object of class marketRisk created using the constructor `marketRisk`.
- `...` additional arguments.

Value
a logical value, is the scenarioRisk consistent with the marketRisk?

See Also
- `check`, `scenarioRisk`.

check.standalone  

Checking Consistency of a Standalone with a MarketRisk

Description
check is a generic S3 method for classes inheriting from item. It is a logical method checking if the item is well defined with respect to a risk (i.e., that all information necessary for valuating the item is available).

Usage
```r
## S3 method for class 'standalone'
check(object, marketRisk, ...)
```
compute

Arguments

object S3 object of class standalone.
market риск S3 object of class marketRisk created using the constructor marketRisk.
... additional arguments.

Value

a logical value, is the standalone consistent with the marketRisk?

See Also

check, standalone.

Description

compute is a generic S3 method for S3 classes inheriting from risk. It returns a vector of aggregated simulations for the corresponding risk.

Usage

compute(object, ...)

Arguments

object an S3 object to compute.
... additional parameters.

Value

results of the computation.
compute.healthRisk: Compute a HealthRisk

Description

compute is a generic S3 method for classes inheriting from risk. It returns a vector of aggregated simulations for the corresponding risk.

Usage

```r
## S3 method for class 'healthRisk'
compute(object, market.risk, health.item, nsim, seed = NULL, ...)
```

Arguments

- `object`: S3 object of class `healthRisk`.
- `market.risk`: S3 object of class `marketRisk` created using `marketRisk`.
- `health.item`: S3 object of class `health` from a portfolio.
- `nsim`: strictly positive integer value of length one. The number of simulations.
- `seed`: positive integer value of length one. The seed for reproducibility.
- `...`: additional arguments.

Value

a `data.table` value containing one column named "healthRisk". The simulations result for a `healthRisk`.

See Also

`compute.healthRisk`.

compute.lifeRisk: Compute a LifeRisk

Description

compute is a generic S3 method for classes inheriting from risk. It returns a vector of aggregated simulations for the corresponding risk.

Usage

```r
## S3 method for class 'lifeRisk'
compute(object, market.risk, life.item, nsim, seed = NULL, ...)
```
compute.macroEconomicScenarios

Arguments

object S3 object of class lifeRisk.
market.risk S3 object of class marketRisk created using the constructor marketRisk.
life.item S3 object of class life from a portfolio.
n.sim strictly positive integer value of length one. The number of simulations.
seed positive integer value of length one. The seed for reproducibility.
... additional arguments.

Value

a data.table value containing one column named "lifeRisk". The simulations result for a lifeRisk.

See Also

compute, lifeRisk.

compute.macroEconomicScenarios

Computing Macro Economic Scenarios

Description

Computing Macro Economic Scenarios

Usage

## S3 method for class 'macroEconomicScenarios'
compute(object, market.risk, portfolio, ...)

Arguments

object an S3 object of class economicScenarios.
market.risk an S3 object of class marketRisk.
portfolio an S3 object of class portfolio.
... additional arguments.

Value

a data.table with the macro economic scenario values.
compute.marketRisk  Compute a MarketRisk

Description

compute is a generic S3 method for classes inheriting from risk. It returns a vector of aggregated simulations for the corresponding risk.

Usage

```r
## S3 method for class 'marketRisk'
compute(object, market.items, standalones = NULL, nsim, seed = NULL, nested.market.computations = F, ...)
```

Arguments

- `object` S3 object of class marketRisk.
- `market.items` list with elements being object of S3 classes inheriting from marketRisk.
- `standalones` list of possible standalones (default NULL).
- `nsim` strictly positive integer value of length one. The number of simulations.
- `seed` positive integer value of length one. The seed for reproducibility.
- `nested.market.computations` logical value of length one, by default set to FALSE. Should the market items valuations be nested by item types?
- `...` additional arguments.

Value

a list of numeric values. The simulation results for a marketRisk.

See Also

compute, marketRisk.

compute.nonLifeRisk  Compute a nonLifeRisk

Description

compute is a generic S3 method for S3 classes inheriting from risk. It returns a vector of aggregated simulations for the corresponding risk.
compute.participationRisk

Usage

```r
## S3 method for class 'nonLifeRisk'
compute(object, nsim, seed = NULL, market.risk, ...)
```

Arguments

- **object**: an S3 object of class `nonLifeRisk`.
- **nsim**: a strictly positive integer value of length one. The number of simulations.
- **seed**: a strictly positive integer value of length one. The seed for reproducibility.
- **market.risk**: an S3 object of class `marketRisk` created using `marketRisk`.
- **...**: additional parameters.

Value

A `data.table` value containing one column named `nonLifeRisk`. The simulations result for a `nonLifeRisk`.

See Also

- `compute, nonLifeRisk`.

compute.participationRisk

*pCompute a participationRisk*

Description

`compute` is a generic S3 method for S3 classes inheriting from `risk`. It returns a vector of aggregated simulations for the corresponding risk.

Usage

```r
## S3 method for class 'participationRisk'
compute(object, market.risk, participation.item, nsim, seed = NULL, ...)
```

Arguments

- **object**: S3 object of class `participationRisk`.
- **market.risk**: S3 object of class `marketRisk`.
- **participation.item**: S3 object of class `participation`.
- **nsim**: strictly positive integer value of length one. The number of simulations.
- **seed**: positive integer value of length one. The seed for reproducibility.
- **...**: additional arguments.
compute.scenarioRisk

Value

a data.table value containing one column named participation. The simulations result for a participationRisk.

See Also

compute, participationRisk, participation.

data.frame

compute.scenarioRisk  Compute a ScenarioRisk

Description

compute is a generic S3 method for classes inheriting from risk. It returns a vector of aggregated simulations for the corresponding risk.

Usage

## S3 method for class 'scenarioRisk'
compute(object, nsim, seed = NULL, marketRisk, ...)

Arguments

object  S3 object of class scenarioRisk.
nsim  strictly positive integer value of length one. The number of simulations.
seed  positive integer value of length one. The seed for reproducibility.
marketRisk  S3 object of class marketRisk created using the constructor marketRisk.
...  additional arguments.

Value

a data.table value containing one column named "scenarioRisk". The simulations result for a scenarioRisk.

See Also

compute, scenarioRisk.
**compute.sstModel**

*Compute a sstModel*

**Description**

Compute method for the S3 class sstModel. It allows to compute (via Monte-Carlo simulations) all risks inherent to an insurer portfolio in the context of the Swiss Solvency Test (explanations on the model can be found in the FINMA technical document "SST-Marktrisiko und -Aggregation Technische Beschreibung". The output of is an S3 object of class sstOutput on which SST figures can be computed.

**Usage**

```r
## S3 method for class 'sstModel'
compute(object, nsim, seed = NULL,
        nested.market.computations = F, ...)
```

**Arguments**

- `object`: S3 object of class sstModel.
- `nsim`: strictly positive integer value of length one. The number of simulations.
- `seed`: positive integer value of length one. The seed for reproducibility.
- `nested.market.computations`: logical value of length one, by default set to FALSE. Should the market items valuations be nested (and saved) by item types?
- `...`: additional arguments.

**Value**

an S3 object, instance of the class sstOutput.

**See Also**

`compute, sstModel`.

---

**computeConstant**

*Compute The Normalizing Constant for a log-Normal Random Variable*

**Description**

This private function allows to compute scaling constants in the valuation formulas.
conditionalReordering

Usage

computeConstant(id, scale, cov.matrix)

Arguments

id an integer value. The risk-factor ids involved in the valuation formula.
scale a numeric value. The scales corresponding to those risk-factors.
cov.matrix a numeric matrix. The covariance matrix of the risk-factors.

Value

A numeric value, the scaling constant. This is equal to \(0.5\) times the variance of the linear combination of the risk-factors provided in the parameters.

coronalReordering Conditional Reordering

Description

function to generate ranks that have been simply reordered with a Gaussian copula or conditionally reordered with Gaussian copula stressed scenarios from a base Gaussian copula.

Usage

conditionalReordering(n, list.correlation.matrix, name,
scenario.probability = NULL, region.boundaries = NULL,
region.probability = NULL, keep.realized.scenario = F)

Arguments

n positive numeric value of length one. The number of ranks to produce (equal to the number of simulations of the model).

list.correlation.matrix list of correlation matrices, the correlation matrix corresponding to the base normal copula should be provided as a named member “base” in the list (and in first position). The rest of the scenarios should be named in the list by a unique identifier that should match the column names of the argument region.boundaries. Please consider that if no scenario correlation matrices are provided, then simple reordering with the “base” correlation matrix is undertaken (note also that in this case, we require scenario.probability, region.boundaries and region.probability to be NULL).

name character value of length between 0 and 4. It should indicate the names of the subset of risks among:

- market
- life
containsHealth

- health
- nonlife

that are aggregated together with the reordering algorithm. The order of risks in this vector should respect the order defined in the correlation matrices in list.correlation.matrix.

scenario.probability

numeric value giving the scenario probabilities (these probabilities should be provided in the same order as the order of scenarios in list.correlation.matrix (following the correlation matrix named "base").

region.boundaries

matrix with named columns and rows giving the thresholds for each regions (boundaries of the scenario rectangles). Each line represents a given scenario and each column a given quantity to reorder. The rownames should match the scenario names and the colnames should match the risks respecting the order prescribed in both name and the colnames of each correlation matrix in list.correlation.matrix.

region.probability

numeric vector giving the probability under the base Gaussian copula (characterized by the correlation matrix named "base") to hit the scenario regions given by each line in regions.boundary.

keep.realized.scenario

logical value. Should we keep the realized scenario for each line?

Value

a data.table with the final ranks (between 0 and 1) with which we should reorder the given simulations.

---

**containsHealth**

*Checks if the object contains a healthRisk.*

**Description**

S3 generic method to check if the object contains a healthRisk.

**Usage**

`containsHealth(object, ...)`

**Arguments**

- `object` an S3 object.
- `...` additional parameters.

**Value**

a logical value.
containsHealth

Description
S3 generic method to check if the object contains a healthRisk.

Usage
## S3 method for class 'sstOutput'
containsHealth(object, ...)

Arguments
- **object**: sstOutput object.
- **...**: additional arguments.

Value
a logical value.

See Also
- `containsHealth`

containsInsurance

Description
S3 generic method to check if the object contains an insuranceRisk.

Usage
containsInsurance(object, ...)

Arguments
- **object**: an S3 object.
- **...**: additional parameters.
Description
S3 generic method to check if the object contains a insuranceRisk.

Usage
## S3 method for class 'sstOutput'
containsInsurance(object, ...)

Arguments

- object: sstOutput object.
- ...: additional arguments.

Value
a logical value.

See Also
containsInsurance.

---

containsLife
Checks if the object contains a lifeRisk.

Description
S3 generic method to check if the object contains a lifeRisk.

Usage
containsLife(object, ...)
containsLife.sstOutput

Arguments

  object an S3 object.
  ... additional parameters.

Value

  a logical value.

See Also

  containsLife.

---

containsLife.sstOutput

  containsLife Helper

Description

S3 generic method to check if the object contains a lifeRisk.

Usage

  ## S3 method for class 'sstOutput'
  containsLife(object, ...)

Arguments

  object sstOutput object.
  ... additional arguments.

Value

  a logical value.

See Also

  containsLife.
containsMarket

Checks if the object contains a MarketRisk.

Description
S3 generic method to check if the object contains a MarketRisk.

Usage
containsMarket(object, ...)

Arguments
- object: an S3 object.
- ...: additional parameters.

Value
a logical value.

See Also
containsMarket.

containsMarket.sstOutput

containsMarket Helper

Description
S3 generic method to check if the object contains a MarketRisk.

Usage

## S3 method for class 'sstOutput'
containsMarket(object, ...)

Arguments
- object: sstOutput object.
- ...: additional arguments.

Value
a logical value.
containsNonLife.sstOutput

See Also
containsMarket.

containsNonLife  Checks if the object contains nonLifeRisk.

Description
S3 generic method to check if the object contains nonLifeRisk.

Usage
containsNonLife(object, ...)

Arguments
object  an S3 object.
...  additional parameters.

Value
a logical value.

See Also
containsNonLife.

containsNonLife.sstOutput  containsNonLife Helper

Description
S3 generic method to check if the object contains nonLifeRisk.

Usage
## S3 method for class 'sstOutput'
containsNonLife(object, ...)

Arguments
object  sstOutput object.
...  additional arguments.
containsParticipation

Value

  a logical value.

See Also

  containsNonLife.

containsParticipation checks if the object contains participation.

Description

  S3 generic method to check if the object contains participation.

Usage

  containsParticipation(object, ...)

Arguments

  object an S3 object.
  ... additional parameters.

Value

  a logical value.

See Also

  containsParticipation.

containsParticipation.sstOutput

containsParticipation Helper

Description

  S3 generic method to check if the object contains participation.

Usage

  ## S3 method for class 'sstOutput'
  containsParticipation(object, ...)

**Arguments**

- **object**: sstOutput object.
- **...**: additional arguments.

**Value**

a logical value.

**See Also**

containsParticipation.

---

**containsScenario**

*Checks if the object contains scenario.*

---

**Description**

S3 generic method to check if the object contains scenario.

**Usage**

`containsScenario(object, ...)`

**Arguments**

- **object**: an S3 object.
- **...**: additional parameters.

**Value**

a logical value.

**See Also**

containsScenario.
containsScenario.sstOutput

containsScenario.sstOutput

containsScenario Helper

Description
S3 generic method to check if the object contains scenario.

Usage
```r
## S3 method for class 'sstOutput'
containsScenario(object, ...)
```

Arguments
- `object`: sstOutput object.
- `...`: additional arguments.

Value
a logical value.

See Also
- `containsScenario`.

creditRisk

Credit risk

Description
S3 generic method to get credit risk.

Usage
```r
creditRisk(object, ...)
```

Arguments
- `object`: an S3 object.
- `...`: additional parameters.

Value
a numeric value.
creditRisk.sstOutput  
*Get Credit Risk from sstOutput*

**Description**

S3 method to extract the credit risk from an sstOutput.

**Usage**

```r
## S3 method for class 'sstOutput'
creditRisk(object, ...)
```

**Arguments**

- `object`  
  S3 object of class sstOutput.
- `...`  
  Additional parameters.

**Value**

a numeric value. The credit risk.

**See Also**

`creditRisk`.  

---

**currency**  
*Constructing a Currency (FX Exchange Rate Risk Factor)*

**Description**

Constructor for the S3 class currency. It allows to define a currency (fx rate) risk factor. This risk factor refers to the "Fremdwährungsrisikofaktors" change $\Delta R_{t,FX_j}$ for a certain index $j$ in the all valuation functions at presented in the FINMA document "SST-Marktrisiko und -Aggregation Technische Beschreibung".

**Usage**

```r
currency(name, from, to)
```
currencyIsIn

Arguments

name
a character value of length one. This corresponds to the name in the covariance matrix of the marketRisk to which the currency risk factor is mapped. This means that the risk factor change $\Delta R_{F_{t,FX_j}}$ in the FINMA document "SST-Marktrisiko und -Aggregation Technische Beschreibung" will be assumed to be modeled by the underlying normal random variable corresponding to name in the covariance matrix.

from
a character value of length one. The starting currency corresponding to the FX index $j$ in the FINMA document "SST-Marktrisiko und -Aggregation Technische Beschreibung".

to
a character value of length one. The arrival currency to which the exchange rate $FX_j$ is mapped.

Value

An S3 object, instance of the class currency.

Note

Please consider that we do not allow for scaled currency risk factors.

Examples

```r
# constructing a currency risk factor
# (assuming "EURCHF" exists in marketRisk).
cur <- currency(name = "EURCHF",
                 from = "EUR",
                 to = "CHF")
```

currencyIsIn

Currency in Object?

Description

S3 generic to check that the currency is in the object.

Usage

```r
currencyIsIn(object, ...)
```

Arguments

object
an S3 object potentially containing the currency.

... additional parameters.
Value

a logical value.

currencyIsIn.standalone

Currency in Standalone?

Description

S3 generic to check that the currency is in the object.

Usage

## S3 method for class 'standalone'
currencyIsIn(object, from, to, ...)

Arguments

object S3 object of class standalone.
from a character value of length one. A currency.
to a character value of length one. A currency.
... additional arguments.

Value

a logical value, is the currency in the standalone?

See Also

currencyIsIn.

delta

Constructing a Delta-Normal Remainder Term with Respect to MarketRisk

Description

delta Constructor for the S3 class delta. It allows to build for the sensitivities with respect to the market risk-factors of the total positions not modelled by the other marketItem classes used in a delta-normal remainder term presented in the FINMA technical document "SST-Marktrisiko und -Aggregation Technische Beschreibung".

Usage

delta(name, currency, sensitivity)
Arguments

name numeric value. The names of the market base risk factors (the base risk factors defined in marketRisk) with respect to which sensitivities are computed (non-zero). This vector should not contain duplicated names.

currency character value representing currencies in which the sensitivities are expressed. If the currency specified does not match the base currency of the marketRisk, the initial fx-rates will be used to convert to the base currency. Nevertheless, it is forced at construction of a portfolio that the sensitivities should be provided in the the portfolio base currency.

sensitivity numeric value giving sensitivities for the corresponding market risk-factors provided in name. These quantities explicitly relates to the "Sensitivität" as defined in the FINMA technical document "SST-Marktrisiko und -Aggregation Technische Beschreibung", you can refer to this document for their estimation procedures. Sensitivities must be expressed in the corresponding currencies, i.e. in currency.

Value

an S3 object, instance of the class delta.

Note

All parameters must be of equal length.

See Also

summary.delta, print.delta.

Examples

# Creating a new delta.
d <- delta(name = c("equity", "2YCHF", "EURCHF"),
currency = c("EUR", "CHF", "EUR"),
sensitivity = c(100, 150, 130))

Description

Constructor for the S3 class equity. It allows to define an equity-type risk factor. This risk factor refers to the "Preisrisikofaktor" change $\Delta RF_{i,t}$ for a certain index $i$ in the valuation function for "Aktiven mit direkt marktabhängigen Preisen" presented in the FINMA document "SST-Marktrisiko und -Aggregation Technische Beschreibung".
Usage

equity(name, type, currency, scale = NULL)

Arguments

name a character value of length one. This corresponds to the name in the covariance matrix of the marketRisk to which the equity risk factor is mapped. This means that the risk factor change $\Delta R_{F,t,i}$ in the FINMA document "SST-Marktrisiko und -Aggregation Technische Beschreibung" will be assumed to be modeled by the underlying normal random variable corresponding to name in the covariance matrix (potentially scaled by scale if not NULL).

type a character value of length one. The type of equity. (e.g. "equity", "hedge fund", etc.). This parameter is a unique identifier of the equity risk factor corresponding to the index $i$ introduced above. The following words are reserved and should not be used:

- currency
- rate
- pcRate
- spread

currency a character value of length one. The currency in which the underlying asset with direct market price ("Aktiv mit direkt marktabhängigen Preisen") is valuated. This refers to the currency corresponding to the index $j$ in the FINMA document "SST-Marktrisiko und -Aggregation Technische Beschreibung".

scale a numeric value of length one. If not set NULL, this defines a scaled risk factor equal to scale times the risk factor defined by name in the covariance matrix contained in marketRisk. By default its value is scale = NULL.

Value

An S3 object, instance of the class equity.

Examples

# constructing a non-scaled equity risk factor
# (assuming "MSCI_CHF" exists in marketRisk).
e <- equity(name = "MSCI_CHF",
    type = "equity",
    currency = "CHF")

# constructing a scaled equity risk factor
# (assuming "MSCI_CHF" exists in marketRisk).
e <- equity(name = "MSCI_CHF",
    type = "equity",
    currency = "CHF",
    scale = 0.5)
**equityIsIn**

---

**equityIsIn**  
*Equity in Object?*

---

**Description**

S3 generic to check that the equity is in the object.

**Usage**

```r
equityIsIn(object, ...)
```

**Arguments**

- `object`  
an S3 object potentially containing the equity.
- `...`  
additional parameters.

**Value**

a logical value.

---

**equityIsIn.standalone**  
*Equity in Standalone?*

---

**Description**

S3 generic to check that the equity is in the object.

**Usage**

```r
## S3 method for class 'standalone'
equityIsIn(object, type, currency, ...)
```

**Arguments**

- `object`  
S3 object of class standalone.
- `type`  
character value of length one. The type of an asset.
- `currency`  
character value of length one. The currency of the asset.
- `...`  
additional arguments.

**Value**

a logical value, is the equity in the standalone?

**See Also**

equityIsIn, asset.
**expectedShortfall**  
*Compute the Expected Shortfall*

**Description**  
Function to compute the alpha-Expected Shortfall of a vector.

**Usage**  
```r  
expectedShortfall(x, alpha = 0.01, sup = F, ...)  
```

**Arguments**  
- `x`  
a numeric vector. The vector from which to compute the expected shortfall.

- `alpha`  
a numeric value. The alpha-Expected Shortfall, must take values between 0 and 1. Please note that `alpha` represents the mass lying below the `alpha` quantile of `x` in the case `sup = FALSE` or the mass lying above the `1-alpha` quantile of `x` in the other case `sup = TRUE`.

- `sup`  
a logical value. If `TRUE` the function returns the upper expected shortfall and otherwise the lower. Default is set to `FALSE`.

- `...`  
additional parameters.

---

**excelToSstModel**  
*Parsing an Excel Template to sstModel*

**Description**  
This function is intended to parse the excel template provided by FINMA into an sstModel.

**Usage**  
```r  
excelToSstModel(path, with.log = F)  
```

**Arguments**  
- `path`  
a character value. A valid path of an input excel workbook. The path can be relative or not.

- `with.log`  
logical value. Should the error/warning-log be returned?

**Value**  
an S3 object of class `sstModel`, built from the input fundamental data sheets.

**See Also**  
[`sstModel`](#).

---
Value

a numeric value. The expected shortfall.

Note

Please consider that we include the boundary value into the empirical mean estimation.

---

**format.asset**  
*Formating an Asset with Direct Market Price*

**Description**

format method for the S3 class asset.

**Usage**

```r
## S3 method for class 'asset'
format(x, ...)
```

**Arguments**

- `x` S3 object of class asset.
- `...` additional arguments.

**Value**

a character value.

**See Also**

`format, asset`.

---

**format.assetForward**  
*Formating an Index-Forward*

**Description**

format method for the S3 class assetForward.

**Usage**

```r
## S3 method for class 'assetForward'
format(x, ...)
```
Arguments

x S3 object of class assetForward.
...
additional arguments.

Value

a character value.

See Also

format, assetForward.

Description

Formating a Fixed-Income-Asset

Usage

## S3 method for class 'cashflow'
format(x, ...)

Arguments

x S3 object of class cashflow.
...
additional arguments.

Value

a character value.

See Also

format, cashflow
### format.delta

**Formating a Delta-Normal Remainder Term**

**Description**
Formating a Delta-Normal Remainder Term

**Usage**
```r
## S3 method for class 'delta'
format(x, ...)
```

**Arguments**
- `x` S3 object of class `delta`
- `...` additional arguments.

**Value**
a character value.

**See Also**
- `format`, `delta`.

---

### format.fxForward

**Formating an FX-Forward**

**Description**
format method for the S3 class `fxForward`.

**Usage**
```r
## S3 method for class 'fxForward'
format(x, ...)
```

**Arguments**
- `x` an S3 object of class `fxForward`
- `...` additional parameters.

**Value**
a character value.
format.health  

**Description**

Formatting a Health Delta-Normal Term

**Usage**

```r
define method for class 'health'
format(x, ...)
```

**Arguments**

- `x` - S3 object of class health.
- `...` - additional arguments.

**Value**

a character value.

**See Also**

format.health.

format.healthRisk  

**Description**

Formatting a HealthRisk

**Usage**

```r
define method for class 'healthRisk'
format(x, ...)
```

**Arguments**

- `x` - S3 object of class healthRisk.
- `...` - additional arguments.
format.liability  

Value

a character value.

See Also

format.healthRisk.

format.liability   Formating an Insurance Liability

Description

Formating an Insurance Liability

Usage

## S3 method for class 'liability'
format(x, ...)

Arguments

x   S3 object of class liability.
...

Value

a character value.

See Also

format.liability.

format.life   Formating a Life Delta-Normal Remainder Term

Description

Formating a Life Delta-Normal Remainder Term

Usage

## S3 method for class 'life'
format(x, ...)


Arguments

- `x`: an S3 object of class `lifeRisk`.
- `...`: additional arguments.

Value

- a character value.

See Also

- `format`, `lifeRisk`.

---

**format.lifeRisk**

*Formating a LifeRisk*

**Description**

Formating a LifeRisk

**Usage**

```r
## S3 method for class 'lifeRisk'
format(x, ...)
```

**Arguments**

- `x`: an S3 object of class `lifeRisk`.
- `...`: additional arguments.

**Value**

- a character value.

**See Also**

- `format.lifeRisk`. 

---
format.marketRisk

Description

format method for S3 class marketRisk.

Usage

## S3 method for class 'marketRisk'
format(x, ...)

Arguments

x S3 object of class marketRisk.
...

Value

a character value.

See Also

format, marketRisk.

format.nonLifeRisk

Description

Formatting a nonLifeRisk

Usage

## S3 method for class 'nonLifeRisk'
format(x, ...)

Arguments

x S3 object of class nonLifeRisk.
...

Value

a character value.
format.participation

Formating a Participation

Description
Formating a Participation

Usage
```r
## S3 method for class 'participation'
format(x, ...)
```

Arguments
- `x` : S3 object of the class participation.
- `...` : additional arguments.

Value
'a' character value.

See Also
- `format.participation`

format.participationRisk

Formating a ParticipationRisk

Description

format method for S3 class participationRisk.

Usage
```r
## S3 method for class 'participationRisk'
format(x, ...)
```

Arguments
- `x` : an S3 object of class participationRisk.
- `...` : additional parameters.

See Also
- `format.participationRisk`
**Value**

a character value.

**See Also**

`format.participationRisk`.

---

**format.portfolio**  
*Formating a Portfolio*

**Description**

Formating a Portfolio

**Usage**

```r
## S3 method for class 'portfolio'
format(x, ...)
```

**Arguments**

- `x`  
  S3 object of class `portfolio`.

- `...`  
  additional arguments.

**Value**

a character value.

**See Also**

`format.portfolio`.

---

**format.scenarioRisk**  
*Formating a ScenarioRisk*

**Description**

Formating a ScenarioRisk

**Usage**

```r
## S3 method for class 'scenarioRisk'
format(x, ...)
```
Arguments

- `x` S3 object of class `scenarioRisk`.
- `...` additional arguments.

Value

- a character value.

See Also

- `format`, `scenarioRisk`.

Description

Formatting a `sstModel`

Usage

```r
# S3 method for class 'sstModel'
format(x, ...)
```

Arguments

- `x` S3 object of class `sstModel`.
- `...` additional arguments.

Value

- a character value.

See Also

- `format`, `sstModel`. 
**format.sstOutput**  
*Formating a sstOutput*

**Description**
Formating a sstOutput

**Usage**

```r
## S3 method for class 'sstOutput'
format(x, ...)
```

**Arguments**

- `x`  
  S3 object of class sstOutput.

- `...`  
  additional arguments.

**Value**

a character value.

**See Also**

`format`.

---

**format.standalone**  
*Formating a standalone*

**Description**
Formating a standalone

**Usage**

```r
## S3 method for class 'standalone'
format(x, ...)
```

**Arguments**

- `x`  
  S3 object of class standalone.

- `...`  
  additional arguments.

**Value**

a character value.
See Also

format.standalone.

format.summary.portfolio

Formatting a Summary of Portfolio

Description

Formatting a Summary of Portfolio

Usage

## S3 method for class 'summary.portfolio'
format(x, ...)

Arguments

x an S3 object of class summary.portfolio.
...
additional parameters.

Value

a character value.

See Also

format

format.summary.sstModel

Formatting a Summary of sstModel

Description

Formatting a Summary of sstModel

Usage

## S3 method for class 'summary.sstModel'
format(x, ...)

Arguments

x an S3 object of class summary.sstModel.
...
additional parameters.
Value

a character value.

See Also

format

---

format.summary.sstOutput

**Formatting a Summary of sstOutput**

**Description**

Formatting a Summary of sstOutput

**Usage**

```r
## S3 method for class 'summary.sstOutput'
format(x, ...)
```

**Arguments**

- `x` S3 object of class summary.sstOutput.
- `...` additional arguments.

**Value**

a character value.

See Also

format.

---

fxForward

**Constructing an FX-Forward**

**Description**

Constructor for the S3 class fxForward. It allows to build for an fx-forward referred under the name "FX-Forward" in the FINMA technical document "SST-Marktrisiko und -Aggregation Technische Beschreibung".

**Usage**

```
fxForward(domestic, foreign, time, nominal, rate, position)
```
Arguments

- **domestic**: character value of length one representing the base currency, i.e. the arrival currency from which foreign fx rates are hedged. This parameter relates to the index $0$ (base currency) in the FINMA document “SST-Marktrisiko und -Aggregation Technische Beschreibung”.

- **foreign**: character value of length one representing the foreign currency, i.e. the currency on which fx rate converting foreign back to domestic is hedged. This parameter relates to the fxForward index $j$ (foreign currency) in the FINMA document “SST-Marktrisiko und -Aggregation Technische Beschreibung”.

- **time**: strictly positive integer value of length one representing the time-to-maturity from $t = 0$. This parameter relates to the fxForward variable $\tau$ in the FINMA document “SST-Marktrisiko und -Aggregation Technische Beschreibung”.

- **nominal**: strictly positive numeric value of length one representing the nominal value of the contract expressed in the foreign currency. This parameter relates to the fxForward quantity $N_j^\tau$ in the FINMA document “SST-Marktrisiko und -Aggregation Technische Beschreibung”.

- **rate**: positive numeric value of length one representing the forward fx rate settled in the contract from currency foreign to currency domestic. This parameter relates to the fxForward quantity $F_{\tau}$ in the FINMA document “SST-Marktrisiko und -Aggregation Technische Beschreibung”.

- **position**: character value of length one. This can be either "long" or "short" according to the definition of long and short forwards in the FINMA document “SST-Marktrisiko und -Aggregation Technische Beschreibung”.

Value

- an S3 object, instance of the class fxForward.

See Also

- `summary.fxForward`, `print.fxForward`.

Examples

```r
# Creating new fxForwards.
fx.forward.1 <- fxForward("USD", "EUR", 1, 1000, 1.05, "long")
fx.forward.2 <- fxForward("CHF", "EUR", 10, 500, 1.1, "short")
```
generateError

Generate error message from an error log

Description
this function transforms an error log into an error message.

Usage
generateError(error.log, warning.log, line.break = "\n")

Arguments
error.log a data.frame with following fields:
  • sheet: character, the sheet name.
  • row: integer, the row position.
  • column: integer, the column position.
  • message: character, the error message.
warning.log a data.frame similar to error.log.
line.break a character value, separation between error messages.

Value
a character value, the corresponding error message.

See Also
excelToSstModel.

generateExpression

Generate an Expression

Description
method to generate an expression.

Usage
generateExpression(object, ...)

Arguments
object an S3 object.
... additional arguments.
**generateExpression.portfolio**

*Generate the Market Valuation Expression for a Portfolio*

**Description**

Method to generate the market valuation expression for a given portfolio and a given subset of item classes.

**Usage**

```r
## S3 method for class 'portfolio'
generateExpression(object, market.item.types, market.risk, standalone = NULL, ...)
```

**Arguments**

- **object**: S3 object of class portfolio.
- **market.item.types**: Character value indicating the item classes for which the market expression should be computed and aggregated, this should be a subset of the following values:
  - asset
  - cashflow
  - liability
  - assetForward
  - fxForward
  - delta
  you can also provide the value "all", in this case all market item expressions in the portfolio are computed and aggregated.
- **market.risk**: S3 object of class marketRisk.
- **standalone**: S3 object of class standalone.
- **...**: Additional arguments.

**Value**

A character value, the market expression.

**Note**

Please consider that the expression are centered (mean zero).
**generateFunction**  
*Generate a Function*

**Description**  
method to generate a function.

**Usage**  

```r
generateFunction(object, ...)
```

**Arguments**

- `object` an S3 object.
- `...` additional arguments.

**Value**

a function.

**generateFunction.portfolio**  
*Generate the Market Valuation Function for a Portfolio*

**Description**  
method to generate the market valuation function for a given portfolio and all positions (including participation if any).

**Usage**

```r
## S3 method for class 'portfolio'
generateFunction(object, market.risk, ...)
```

**Arguments**

- `object` S3 object of class portfolio.
- `market.risk` S3 object of class marketRisk.
- `...` additional arguments.
getCurrencyId

Value

a function, the market valuation function with the following parameter:

- \( x \): a matrix of simulation with named columns corresponding exactly to the name of base-risk factors in \( \text{marketRisk} \) keeping the same order or an unnamed vector of simulations keeping the same ordering of risk factors as in the covariance matrix defined in \( \text{marketRisk} \). Please note that if the portfolio contains a participation, then an additional column (in the case of matrix input) named participation or an additional entry (in the case of vector input) should be provided in the last position.

Note

Please note that the valuation functions here are not centered.

See Also

portfolio.

---

**getDescription**

Get A Currency ID

Description

S3 generic to get a currency id.

Usage

getchCurrencyId(object, ...)

Arguments

  object an S3 object containing the currency.

  ... additional parameters.

Value

a numeric value.
getCurrencyId$marketRisk

---

**getCurrencyId$marketRisk**

*Get A Currency ID*

---

**Description**

S3 generic to get a currency id.

**Usage**

```r
## S3 method for class 'marketRisk'
getCurrencyId(object, from, to, ...)
```

**Arguments**

- **object**: S3 object of class marketRisk.
- **from**: character value. A well defined type in object.
- **to**: character value. A well defined currency in object for the departure currency from.
- **...**: additional parameters.

**Value**

a numeric value.

**Note**

This method is private and does not test validity or coherence of its arguments.

**See Also**

`getCurrencyId`.

---

getCurrencyName

---

**getCurrencyName**

*Get A Currency Name*

---

**Description**

S3 generic to get a currency name.

**Usage**

```r
generateCurrencyName(object, ...)```

getCurrencyName.marketRisk

Get A Currency Name

Arguments

object an S3 object containing the currency.
... additional parameters.

Value

a character value.

Description

S3 generic to get a currency name.

Usage

## S3 method for class 'marketRisk'
getCurrencyName(object, from, to, ...)

Arguments

object S3 object of class marketRisk.
from character value. A well defined type in object.
to character value. A well defined currency in object for the departure currency from.
... additional parameters.

Value

a character value.

Note

This method is private and does not test validity or coherence of its arguments.

See Also

getCurrencyName.
\textbf{getCurrencyScale}  \hspace{1cm}  \textit{Get A Currency Scale}

\underline{Description}  
S3 generic to get a currency scale.

\underline{Usage}  
\texttt{getCurrencyScale(object, \ldots)}

\underline{Arguments}  
\begin{itemize}  
  \item \texttt{object}  \hspace{1cm} an S3 object containing the currency.  
  \item \ldots  \hspace{1cm} additional parameters.  
\end{itemize}

\underline{Value}  
a numeric value.

\underline{getCurrencyScale.marketRisk}  \hspace{1cm}  \textit{Get A Currency Scale}

\underline{Description}  
S3 generic to get a currency scale.

\underline{Usage}  
\# S3 method for class 'marketRisk'  
getCurrencyScale(object, from, to, \ldots)

\underline{Arguments}  
\begin{itemize}  
  \item \texttt{object}  \hspace{1cm} S3 object of class marketRisk.  
  \item \texttt{from}  \hspace{1cm} character value. A well defined type in object.  
  \item \texttt{to}  \hspace{1cm} character value. A well defined currency in object for the departure currency from.  
  \item \ldots  \hspace{1cm} additional parameters.  
\end{itemize}

\underline{Value}  
a numeric value.
### getDeltaId

**Get A Delta ID**

**Description**

S3 generic to get a delta id.

**Usage**

```r
getDeltaId(object, ...)
```

**Arguments**

- `object` an S3 object containing the delta.
- `...` additional parameters.

**Value**

a numeric value.

### getDeltaId.marketRisk

**Get A Delta ID**

**Description**

S3 generic to get a delta id.

**Usage**

```r
## S3 method for class 'marketRisk'
getDeltaId(object, name, ...)
```

**Arguments**

- `object` S3 object of class marketRisk.
- `name` character value. A well defined risk factor names in `object`.
- `...` additional parameters.
getDrbc

Value
a numeric value.

Note
This method is private and does not test validity or coherence of its arguments.

See Also
getDeltaId.

getDrbc

S3 generic method to get drbc

Usage
getDrbc(object, with.scenario = F, ...)

Arguments

object an S3 object.
with.scenario a logical value.
... additional parameters.

Value
a numeric value.

getDrbc.sstOutput

S3 generic method to get drbc

Usage

## S3 method for class 'sstOutput'
getDrbc(object, with.scenario = F, exp.shortfall = F, ...

...
Arguments

- **object**: S3 object of class sstOutput.
- **with.scenario**: logical value.
- **exp.shortfall**: logical value, by default set to FALSE. Should the expected shortfall be returned?
- ... additional arguments.

Value

- a numeric value.

See Also

getCode.

---

**getEquityId**  
*Get An Equity ID*

Description

S3 generic to get an equity id.

Usage

getEquityId(object, ...)

Arguments

- **object**: an S3 object containing the equity.
- ... additional parameters.

Value

- a numeric value.
**getEquityId.marketRisk**

*Get An Equity ID*

---

**Description**

S3 generic to get an equity id.

**Usage**

```r
## S3 method for class 'marketRisk'
getEquityId(object, type, currency, ...)
```

**Arguments**

- `object` S3 object of class `marketRisk`.
- `type` character value. A well defined type in `object`.
- `currency` character value. A well defined currency in `object` for the type `type`.
- `...` additional parameters.

**Value**

a numeric value.

**Note**

This method is private and does not test validity or coherence of its arguments.

**See Also**

`getEquityId`.

---

**getEquityName**

*Get An Equity Name*

---

**Description**

S3 generic to get an equity name.

**Usage**

`getEquityName(object, ...)`
getEquityName.marketRisk

Get An Equity Name

Description

S3 generic to get an equity name.

Usage

```r
## S3 method for class 'marketRisk'
getEquityName(object, type, currency, ...)
```

Arguments

- `object`: S3 object of class marketRisk.
- `type`: character value. A well defined type in `object`.
- `currency`: character value, well defined currency in `object` for the type `type`.
- `...`: additional parameters.

Value

- a character value.

Note

This method is private and does not test validity or coherence of its arguments.

See Also

`getEquityName`
getEquityScale  

**Description**

S3 generic to get an equity Scale.

**Usage**

getEquityScale(object, ...)

**Arguments**

- object: an S3 object containing the equity.
- ...: additional parameters.

**Value**

a numeric value.

getEquityScale.marketRisk

**Description**

S3 generic to get an equity Scale.

**Usage**

## S3 method for class 'marketRisk'

getEquityScale(object, type, currency, ...)

**Arguments**

- object: S3 object of class marketRisk.
- type: character value. A well-defined type in object.
- currency: character value. A well-defined currency in object for the type type.
- ...: additional parameters.

**Value**

a numeric value.
**getHealthQuantile**

**Note**

This method is private and does not test validity or coherence of its arguments.

**See Also**

getEquityScale.

---

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getHealthId</td>
<td>Get A Health Item ID</td>
</tr>
<tr>
<td>getHealthQuantile</td>
<td>Get A Health Item Quantile</td>
</tr>
</tbody>
</table>

### getHealthId

**Description**

S3 generic to get a health item id.

**Usage**

```r
getHealthId(object, ...)  
```

**Arguments**

- `object`: an S3 object containing the health item.
- `...`: additional parameters.

**Value**

a numeric value.

### getHealthQuantile

**Description**

S3 generic to get a health item quantile.

**Usage**

```r
getHealthQuantile(object, ...)  
```

**Arguments**

- `object`: an S3 object containing the health item.
- `...`: additional parameters.

**Value**

a numeric value.
**getHealthRisk**

## Description

S3 generic method to get health insurance risk.

## Usage

```r
getHealthRisk(object, ...)  
```

## Arguments

- **object**: an S3 object.
- **...**: additional parameters.

## Value

A numeric value.

---

**getHealthRisk.sstOutput**

## Description

S3 generic method to get health insurance risk.

## Usage

```r  
## S3 method for class 'sstOutput'
getHealthRisk(object, exp.shortfall = F, ...)  
```

## Arguments

- **object**: S3 object of class sstOutput.
- **exp.shortfall**: logical value, by default set to FALSE. Should the expected shortfall be returned?
- **...**: additional arguments.

## Value

A numeric value.

## See Also

`getInsuranceRisk`. 

getInitialFX  

Get An Initial FX

Description
S3 generic to get initial fx.

Usage
getInitialFX(object, ...)

Arguments

object an S3 object containing initial fx.
... additional parameters.

Value
a numeric value.

generic getInitialFX

getInitialFX  

getInitialFX

Get An Initial FX

Description
S3 generic to get initial fx.

Usage
getInitialFX(object, from, to, ...)

Arguments

object S3 object of class marketRisk.
from character value. A well-defined currency defined in object.
to character value. A well-defined currency defined in object.
... additional parameters.

Value
a numeric value.
getInitialRate

Note
This method is private and does not test validity or coherence of its arguments.

See Also
getInitialFX.

generic funtion to get initial rate.

Usage
getInitialRate(object, ...)

Arguments

object an S3 object containg initial rate.
... additional parameters.

Value

a numeric value.

generic funtion to get initial rate.

Usage
## S3 method for class 'marketRisk'
getInitialRate(object, time, currency, ...)

Arguments

object object of class marketRisk.
time integer value. A well defined time in object for currency.
currency character value. A well defined currency in object.
... additional parameters.
**getInsuranceRisk**

**Value**

a numeric value.

**Note**

This method is private and does not test validity or coherence of its arguments.

**See Also**

`getInitialRate`.

---

**getInitialSpread**  
*Get An Initial Spread*

**Description**

S3 generic to get initial spread.

**Usage**

`getInitialSpread(object, ...)`

**Arguments**

- `object`: an S3 object containing initial spread.
- `...`: additional parameters.

**Value**

a numeric value.

---

**getInsuranceRisk**  
*Get Insurance Risk*

**Description**

S3 generic method to get insurance risk.

**Usage**

`getInsuranceRisk(object, ...)`

**Arguments**

- `object`: an S3 object.
- `...`: additional parameters.
getInsuranceRisk.sstOutput

Value

a numeric value.

---

getInsuranceRisk.sstOutput

Get Insurance Risk

Description

S3 generic method to get insurance risk.

Usage

```r
## S3 method for class 'sstOutput'
getInsuranceRisk(object, exp.shortfall = F, ...)
```

Arguments

- `object`: S3 object of class `sstOutput`.
- `exp.shortfall`: logical value, by default set to `FALSE`. Should the expected shortfall be returned?
  - ... additional arguments.

Value

a numeric value.

See Also

- `getInsuranceRisk`

---

getLifeId

Get A Life Item ID

Description

S3 generic to get a life item id.

Usage

```r
getLifeId(object, ...)
```

Arguments

- `object`: an S3 object containing the life item.
- ... additional parameters.
**getLifeId.lifeRisk**  
*Get LifeRisk ID*

**Description**
This method is private and does not test validity or coherence of its arguments.

**Usage**
```r
# S3 method for class 'lifeRisk'
getLifeId(object, name, ...)
```

**Arguments**
- **object**: an S3 object of class lifeRisk.
- **name**: a character value. A well defined risk factor names in object.
- **...**: additional arguments.

**Value**
a numeric value.

**See Also**
- `getLifeId.lifeRisk`.

**getLifeQuantile**  
*Get A Life Item Quantile*

**Description**
S3 generic to get a life item quantile.

**Usage**
```r
generateLifeQuantile(object, ...)
```

**Arguments**
- **object**: an S3 object containing the life item.
- **...**: additional parameters.

**Value**
a numeric value.
getLifeQuantile.lifeRisk

Get LifeRisk Quantiles

Description
This method is private and does not test validity or coherence of its arguments.

Usage
## S3 method for class 'lifeRisk'
getLifeQuantile(object, name, ...)

Arguments
- object: an S3 object of class lifeRisk.
- name: a character value. A well defined risk factor names in object.
- ...: additional arguments.

Value
a numeric value.

See Also
- getLifeQuantile, lifeRisk.

getLifeRisk

Get Life Risk

Description
S3 generic method to get life insurance risk.

Usage
gLifeRisk(object, ...)

Arguments
- object: an S3 object.
- ...: additional parameters.

Value
a numeric value.
getLifeRisk.sstOutput  Get Life Insurance Risk

Description
S3 generic method to get life insurance risk.

Usage
## S3 method for class 'sstOutput'
getLifeRisk(object, exp.shortfall = F, ...)

Arguments
- object: S3 object of class sstOutput.
- exp.shortfall: logical value, by default set to FALSE. Should the expected shortfall be returned?
- ...: additional arguments.

Value
a numeric value.

See Also
getInsuranceRisk.

getMappingTime  Get A Time Mapping

Description
S3 generic to get a time mapping.

Usage
getMappingTime(object, ...)

Arguments
- object: an S3 object containing a time mapping.
- ...: additional parameters.

Value
a character value.
**getMappingTime.marketRisk**

*Get A Time Mapping*

**Description**

S3 generic to get a time mapping.

**Usage**

```r
## S3 method for class 'marketRisk'
getMappingTime(object, time, ...)
```

**Arguments**

- `object` S3 object of class `marketRisk`.
- `time` integer value. A well defined time in `object`.
- `...` additional parameters.

**Value**

a numeric value.

**Note**

This method is private and does not test validity or coherence of its arguments.

**See Also**

`getMappingTime`.

**getMarketParticipationRisk**

*Get Aggregated Market Risk and Participation*

**Description**

S3 generic method to get aggregated market risk and participation.

**Usage**

```r
getMarketParticipationRisk(object, ...)
```
Arguments

object    an S3 object.
...      additional parameters.

Value

a numeric value.

---

getMarketParticipationRisk.sstOutput

*Get Aggregated Market and Participation Risk*

Description

S3 generic method to get aggregated market risk and participation.

Usage

```r
## S3 method for class 'sstOutput'
getMarketParticipationRisk(object, exp.shortfall = F, ...)
```

Arguments

object    S3 object of class sstOutput.
extp.shortfall    logical value, by default set to FALSE. Should the expected shortfall be returned?
...      additional arguments.

Value

a numeric value.

See Also

`getMarketRisk`. 
### getMarketRisk

**Get Market Risk**

#### Description

S3 generic method to get market risk.

#### Usage

```r
getMarketRisk(object, ...)  
```

#### Arguments

- `object`: an S3 object.
- `...`: additional parameters.

#### Value

a numeric value.

---

### getMarketRisk.sstOutput

**Get Market Risk**

#### Description

S3 generic method to get market risk.

#### Usage

```r
## S3 method for class 'sstOutput'
getMarketRisk(object, exp.shortfall = F, ...)  
```

#### Arguments

- `object`: S3 object of class sstOutput.
- `exp.shortfall`: logical value, by default set to `FALSE`. Should the expected shortfall be returned?
- `...`: additional arguments.

#### Value

a numeric value.

#### See Also

`getMarketRisk`. 
getNonLifeRisk

Get nonLife Risk

Description
S3 generic method to get non-life insurance risk.

Usage
getNonLifeRisk(object, ...)

Arguments
object an S3 object.
... additional parameters.

Value
a numeric value.

getNonLifeRisk.sstOutput

Get Non Life Insurance Risk

Description
S3 generic method to get non life insurance risk.

Usage
## S3 method for class 'sstOutput'
getNonLifeRisk(object, exp.shortfall = F, ...)

Arguments
object S3 object of class sstOutput.
exp.shortfall logical value, by default set to FALSE. Should the expected shortfall be returned?
... additional arguments.

Value
a numeric value.

See Also
getInsuranceRisk.
getParticipation

---

**getParticipation**

*Get Participation*

**Description**

S3 generic method to get participation.

**Usage**

```r
getParticipation(object, ...)
```

**Arguments**

- `object`: an S3 object.
- `...`: additional parameters.

**Value**

a numeric value.

---

**getParticipation.sstOutput**

*Get Participation*

**Description**

S3 generic method to get participation.

**Usage**

```r
## S3 method for class 'sstOutput'
getParticipation(object, exp.shortfall = F, ...)
```

**Arguments**

- `object`: S3 object of class sstOutput.
- `exp.shortfall`: logical value, by default set to FALSE. Should the expected shortfall be returned?
- `...`: additional arguments.

**Value**

a numeric value.

**See Also**

`getScenarioRisk`. 

---
getRateId

Get A Rate ID

Description

S3 generic to get a rate id.

Usage

getRateId(object, ...)

Arguments

- object: an S3 object containing the rate.
- ...: additional parameters.

Value

a numeric value.

generateId

Get A Rate ID

Description

S3 generic to get a rate id.

Usage

## S3 method for class 'marketRisk'
getRateId(object, currency, horizon, ...)

Arguments

- object: S3 object of class marketRisk.
- currency: character value. A well-defined currency in object.
- horizon: character value. A well-defined horizon in object for the departure currency.
- ...: additional parameters.

Value

a numeric value.
getRateName

Note

This method is private and does not test validity or coherence of its arguments.

See Also

getRateId.

---

getRateName Get A Rate Name

Description

S3 generic to get a rate name.

Usage

getRateName(object, ...)

Arguments

- object: an S3 object containing the rate.
- ...: additional parameters.

Value

A character value.

---

getRateName.marketRisk Get A Rate Name

Description

S3 generic to get a rate name.

Usage

## S3 method for class 'marketRisk'

getRateName(object, currency, horizon, ...)

Arguments

- object: S3 object of class marketRisk.
- currency: character value. A well defined currency in object.
- horizon: character value. A well defined horizon in object for the departure currency.
- ...: additional parameters.
getRateScale

getRateScale

getRateScale

getRateScale

getRateScale

getRateScale

getRateScale

getRateScale

getRateScale

getRateScale

getRateScale

getRateScale

getRateScale

Value

a character value.

Note

This method is private and does not test validity or coherence of its arguments.

See Also

getRateName.

getRateScale

Get A Rate Scale

Description

S3 generic to get a rate scale.

Usage

generateScale(object, ...)  

Arguments

object  
an S3 object containing the rate.

...  
additional parameters.

Value

a numeric value.

getRateScale

Get A Rate Scale

Description

S3 generic to get a rate scale.

Usage

## S3 method for class 'marketRisk'

generateScale(object, currency, horizon, ...)

getRateScale

Get A Rate Scale

Description

S3 generic to get a rate scale.

Usage

## S3 method for class 'marketRisk'

generateScale(object, currency, horizon, ...)
getScenarioRisk

Arguments

- object: S3 object of class marketRisk.
- currency: character value. A well defined currency in object.
- horizon: character value. A well defined horizon in object for the departure currency.
- ...: additional parameters.

Value

a numeric value.

Note

This method is private and does not test validity or coherence of its arguments.

See Also

getRateScale.

getScenarioRisk  Get Scenario Risk

Description

S3 generic method to get scenario risk.

Usage

getScenarioRisk(object, ...)

Arguments

- object: an S3 object.
- ...: additional parameters.

Value

a numeric value.
getScenarioRisk.sstOutput

Get Scenario Risk

Description
S3 generic method to get scenario risk.

Usage
```r
## S3 method for class 'sstOutput'
getScenarioRisk(object, ...)
```

Arguments
- `object` S3 object of class sstOutput.
- `...` additional arguments.

Value
a numeric value.

See Also
getScenarioRisk.

getSpreadId

Get A Spread ID

Description
S3 generic to get a spread id.

Usage
```r
getSpreadId(object, ...)
```

Arguments
- `object` an S3 object containing the spread.
- `...` additional parameters.

Value
a numeric value.
## getSpreadId

### Description

S3 generic to get a spread id.

### Usage

```r
## S3 method for class 'marketRisk'
getSpreadId(object, currency, rating, ...)
```

### Arguments

- **object**: an S3 object of class `marketRisk`.
- **currency**: a character value. A well defined currency in `object`.
- **rating**: a character value. A well defined rating in `object` for the currency `currency`.
- **...**: additional parameters.

### Value

A numeric value.

### Note

This method is private and does not test validity or coherence of its arguments.

### See Also

- `getSpreadId`

## getSpreadName

### Description

S3 generic to get a spread name.

### Usage

```r
getSpreadName(object, ...)
```
getSpreadName.marketRisk

Get A Spread Name

Description

S3 generic to get a spread name.

Usage

```r
getSpreadName(object, currency, rating, ...)  
```

Arguments

- `object` an S3 object containing the spread.
- `...` additional parameters.

Value

a character value.

Note

This method is private and does not test validity or coherence of its arguments.

See Also

getSpreadName.
getSpreadScale  

**Description**

S3 generic to get a spread scale.

**Usage**

getSpreadScale(object, ...)

**Arguments**

- object: an S3 object containing the spread.
- ...: additional parameters.

**Value**

A numeric value.

---

getSpreadScale.marketRisk

**Description**

S3 generic to get a spread scale.

**Usage**

```r
## S3 method for class 'marketRisk'
getSpreadScale(object, currency, rating, ...)
```

**Arguments**

- object: S3 object of class marketRisk.
- currency: character value. A well-defined currency in object.
- rating: character value. A well-defined rating in object for the currency currency.
- ...: additional parameters.

**Value**

A numeric value.
**health**

**Note**

This method is private and does not test validity or coherence of its arguments.

**See Also**

getSpreadScale.

---

**health**  
*Constructing a Health Delta-Normal Term with Respect to healthRisk*

**Description**

**health** is the constructor for the S3 class health. It allows to build for the sensitivities (understood as volatilities) for health insurance risks.

**Usage**

health(name, currency, sensitivity)

**Arguments**

- **name** character value. the names of the health risk factors. Note that no duplicated names should appear.
- **currency** character value. The currencies in which sensitivity are expressed.
- **sensitivity** positive numeric value. The sensitivities with respect for the corresponding risk-factors. Sensitivities must be expressed in the corresponding currency in the column currency. Nevertheless, it is forced at construction of a portfolio that the sensitivities should be provided in the portfolio base currency. Please note that the sensitivities are understood as volatilities for the the corresponding risks, we thus force the sensitivities to be strictly positive.

**Value**

An S3 object, instance of the class health.

**Note**

All parameters must be of equal length.

**See Also**

summary.health, print.health.
Examples

# Creating a new health.
health1 <- health(name = c("pandemy", "longetivity", "storno"),
                   currency = c("EUR", "CHF", "EUR"),
                   sensitivity = c(100, 150, 130))

Description

healthRisk is the constructor for the S3 class healthRisk. It allows to build for health insurance risks parameters.

Usage

healthRisk(corr.mat)

Arguments

corr.mat matrix of numeric values. It must be a valid correlation matrix. This matrix must have names, i.e. attributes colnames and rownames indicating the names of the corresponding health insurance risk factors.

Value

an S3 object, instance of the class healthRisk.

See Also

summary.healthRisk, print.healthRisk, compute.healthRisk.

Examples

# Creating new healthRisks.

corr.mat <- diag(rep(1, 2))
colnames(corr.mat) <- c("invalidity", "longetivity")
rownames(corr.mat) <- colnames(corr.mat)

healthRisk1 <- healthRisk(corr.mat = corr.mat)
initialFX

Constructing initial FX Rates

Description
Constructor for initial FX values.

Usage
initialFX(from, to, fx)

Arguments
- from: character value, the currencies.
- to: character value, the currencies.
- fx: numeric value, the fx rates.

Value
a data.frame with option stringsAsFactors = FALSE.

See Also
marketRisk.

initialRate

Constructing Initial Interest Rates

Description
Constructor for initial Initial Rates values.

Usage
initialRate(time, currency, rate)

Arguments
- time: integer value, the times to maturity.
- currency: character value, the currencies.
- rate: numeric value, the interest rates.

Value
a data.frame with option stringsAsFactors = FALSE.
See Also

marketRisk.

initialSpread  

Compute Initial Spread

Description

compute initial spread of a bond from its market value.

Usage

initialSpread(market.value, times, coupons, risk.free, ...)

Arguments

market.value  a numeric value, the total market value for the bond.
times  a numeric vector, the times of the coupons.
coupons  a numeric vector, the corresponding coupon cash flows.
risk.free  a numeric vector, the corresponding risk-free rates with continuous compounding.
...  additional parameters to be passed to newtonRaphson.

Value

a numeric value, the corresponding spread.

intToGroups  

Ordered Vector of Integers to List of consecutive integers

Description

this helper function helps to group consecutive integers in a sequence of integers.

Usage

intToGroups(x)

Arguments

x  a vector of integers.

Value

a list of integer vectors.
is.asset  

*Assess Class Membership (asset S3 class)*

**Description**

Functions to test inheritance relationships.

**Usage**

```
is.asset(x)
```

**Arguments**

- `x`: an S3 object.

**Value**

A logical value which indicates membership of class `asset`.

**See Also**

`asset`

---

is.assetForward  

*Assess Class Membership (assetForward S3 class)*

**Description**

Functions to test inheritance relationships.

**Usage**

```
is.assetForward(x)
```

**Arguments**

- `x`: an S3 object.

**Value**

A logical value which indicates membership of class `assetForward`.

**See Also**

`assetForward`
is.cashflow

Assess Class Membership (cashflow S3 class)

Description
Functions to test inheritance relationships.

Usage
is.cashflow(x)

Arguments
x an S3 object.

Value
a logical value which indicates membership of class cashflow.

See Also
cashflow.

is.currency

Assess Class Membership (currency S3 class)

Description
Functions to test inheritance relationships.

Usage
is.currency(x)

Arguments
x an S3 object.

Value
a logical value which indicates membership of class currency.

See Also
is.riskFactor.
**is.delta**  
*Assess Class Membership (delta S3 class)*

**Description**
Functions to test inheritance relationships.

**Usage**

```r
is.delta(x)
```

**Arguments**

- `x`  
an S3 object.

**Value**

a logical value which indicates membership of class `delta`.

**See Also**

`delta`.  

---

**is.equity**  
*Assess Class Membership (equity S3 class)*

**Description**
Functions to test inheritance relationships.

**Usage**

```r
is.equity(x)
```

**Arguments**

- `x`  
an S3 object.

**Value**

a logical value which indicates membership of class `equity`.

**See Also**

`is.riskFactor`.  

is.fxForward

Assess Class Membership (fxForward S3 class)

Description
Functions to test inheritance relationships.

Usage
is.fxForward(x)

Arguments
x an S3 object.

Value
a logical value which indicates membership of class fxForward.

See Also
fxForward.

is.health

Assess Class Membership (health S3 class)

Description
Functions to test inheritance relationships.

Usage
is.health(x)

Arguments
x an S3 object.

Value
a logical value which indicates membership of class health.

See Also
health.
is.\texttt{healthRisk} \hspace{1cm} \textit{Assess Class Membership (healthRisk S3 class)}

\section*{Description}

Functions to test inheritance relationships.

\section*{Usage}

\texttt{is.\texttt{healthRisk}(x)}

\section*{Arguments}

\texttt{x} \hspace{1cm} an S3 object.

\section*{Value}

a logical value which indicates membership of class \texttt{healthRisk}.

\section*{See Also}

\texttt{healthRisk}.

is.\texttt{insuranceItem} \hspace{1cm} \textit{Assess Class Membership (insuranceItem S3 class)}

\section*{Description}

Functions to test inheritance relationships.

\section*{Usage}

\texttt{is.\texttt{insuranceItem}(x)}

\section*{Arguments}

\texttt{x} \hspace{1cm} an S3 object.

\section*{Value}

a logical value which indicates membership of class \texttt{insuranceItem}.
is.insuranceRisk

Assess Class Membership (insuranceRisk S3 class)

Description

Functions to test inheritance relationships.

Usage

is.insuranceRisk(x)

Arguments

x an S3 object.

Value

a logical value which indicates membership of class insuranceRisk.

is.item

Assess Class Membership (item S3 class)

Description

Function to test inheritance relationships.

Usage

is.item(x)

Arguments

x an S3 object.

Value

a logical value which indicates membership of class item.
is.\textit{liability} \hspace{0.5cm} \textit{Assess Class Membership (liability S3 class)}

\textbf{Description}

Functions to test inheritance relationships.

\textbf{Usage}

\texttt{is.\textit{liability}(x)}

\textbf{Arguments}

\texttt{x} \hspace{0.5cm} \text{an S3 object.}

\textbf{Value}

a logical value which indicates membership of class \textit{liability}.

\textbf{See Also}

\texttt{liability}

\textit{is.life} \hspace{0.5cm} \textit{Assess Class Membership (life S3 class)}

\textbf{Description}

Functions to test inheritance relationships.

\textbf{Usage}

\texttt{is.life(x)}

\textbf{Arguments}

\texttt{x} \hspace{0.5cm} \text{an S3 object.}

\textbf{Value}

a logical value which indicates membership of class \textit{life}.

\textbf{See Also}

\texttt{life}. 

is.lifeRisk

Assess Class Membership (lifeRisk S3 class)

Description

Functions to test inheritance relationships.

Usage

is.lifeRisk(x)

Arguments

x an S3 object.

Value

a logical value which indicates membership of class lifeRisk.

See Also

liferisk.

is.macroEconomicScenarios

Assess Class Membership (macroEconomicScenarios S3 class)

Description

Functions to test inheritance relationships.

Usage

is.macroEconomicScenarios(x)

Arguments

x an S3 object.

Value

a logical value which indicates membership of class macroEconomicScenarios.
is.mappingTable  Assess Class Membership (mappingTable S3 class)

Description
 Functions to test inheritance relationships.

Usage

`is.mappingTable(x)`

Arguments

- `x` an S3 object.

Value

a logical value which indicates membership of class mappingTable.

See Also

`currency, rate, spread, equity, pcRate`.

is.marketItem  Assess Class Membership (marketItem S3 class)

Description

Function to test inheritance relationships.

Usage

`is.marketItem(x)`

Arguments

- `x` an S3 object.

Value

a logical value which indicates membership of class marketItem.
is.marketRisk  

Assess Class Membership (marketRisk S3 class)

Description

Functions to test inheritance relationships.

Usage

is.marketRisk(x)

Arguments

x an S3 object.

Value

a logical value which indicates membership of class marketRisk.

See Also

marketRisk.

is.nonLifeRisk  

Assess Class Membership (nonLifeRisk S3 class)

Description

Functions to test inheritance relationships.

Usage

is.nonLifeRisk(x)

Arguments

x an S3 object.

Value

a logical value which indicates membership of class nonLifeRisk.

See Also

nonLifeRisk.
**is.participation**  
*Assess Class Membership (participation S3 class)*

**Description**  
Functions to test inheritance relationships.

**Usage**  
`is.participation(x)`

**Arguments**  

- `x`: an S3 object.

**Value**  
a logical value which indicates membership of class `participation`.

**See Also**  
`participation`.

---

**is.participationRisk**  
*Assess Class Membership (standalone S3 class)*

**Description**  
Functions to test inheritance relationships.

**Usage**  
`is.participationRisk(x)`

**Arguments**  

- `x`: an S3 object.

**Value**  
a logical value which indicates membership of class `participationRisk`. 
is.pcRate

Assess Class Membership (pcRate S3 class)

Description
Functions to test inheritance relationships.

Usage
is.pcRate(x)

Arguments
x  an S3 object.

Value
a logical value which indicates membership of class pcRate.

See Also
is.riskFactor.

is.portfolio

Assess Class Membership (portfolio S3 class)

Description
Functions to test inheritance relationships.

Usage
is.portfolio(x)

Arguments
x  an S3 object.

Value
a logical value which indicates membership of class portfolio.

See Also
portfolio.
is.rate

Assess Class Membership (rate S3 class)

Description
Functions to test inheritance relationships.

Usage
is.rate(x)

Arguments
x an S3 object.

Value
a logical value which indicates membership of class rate.

See Also
is.riskFactor.

is.risk

Assess Class Membership (risk S3 class)

Description
Functions to test inheritance relationships.

Usage
is.risk(x)

Arguments
x an S3 object.

Value
a logical value which indicates membership of class risk.
**is.riskFactor**

*Assess Class Membership (riskFactor S3 class)*

---

**Description**

Functions to test inheritance relationships.

**Usage**

```r
is.riskFactor(x)
```

**Arguments**

- `x` an S3 object.

**Value**

a logical value which indicates membership of class `riskFactor`.

**See Also**

- `currency`, `rate`, `spread`, `equity`, `pcRate`.

---

**is.scenarioRisk**

*Assess Class Membership (scenerioRisk S3 class)*

---

**Description**

Functions to test inheritance relationships.

**Usage**

```r
is.scenarioRisk(x)
```

**Arguments**

- `x` an S3 object.

**Value**

a logical value which indicates membership of class `scenerioRisk`.

**See Also**

- `scenarioRisk`.
is.sstModel

is.spread

Assess Class Membership (spread S3 class)

Description

Functions to test inheritance relationships.

Usage

is.spread(x)

Arguments

x an S3 object.

Value

a logical value which indicates membership of class spread.

See Also

is.riskFactor.

is.sstModel

Assess Class Membership (sstModel S3 class)

Description

Functions to test inheritance relationships.

Usage

is.sstModel(x)

Arguments

x an S3 object.

Value

a logical value which indicates membership of class sstModel.

See Also

sstModel.
is.sstOutput 

Assess Class Membership (sstOutput S3 class)

Description

Functions to test inheritance relationships.

Usage

is.sstOutput(x)

Arguments

x an S3 object.

Value

a logical value which indicates membership of class sstOutput.

See Also

currency, rate, spread, equity, pcRate.
itemListToExpression  
*Item List to Valuation Expression Helper*

**Description**

helper function to convert a list of market.items to an aggregated valuation expression.

**Usage**

```r
itemListToExpression(item.list, market.item.types, market.risk, standalone = NULL)
```

**Arguments**

- `item.list`: a list of marketItem S3 objects.
- `market.item.types`: a character value representing a subset of marketItem classes.
- `market.risk`: a marketRisk S3 object.
- `standalone`: an S3 object of class standalone.

**Value**

a character value representing the aggregated valuation expression.

---

itemListToFunction  
*Item List to Valuation Function Helper*

**Description**

helper function to convert a list of market.items to an aggregated valuation function.

**Usage**

```r
itemListToFunction(item.list, market.item.types, market.risk, with.constant = F)
```

**Arguments**

- `item.list`: a list of marketItem S3 objects.
- `market.item.types`: a character value representing a subset of marketItem classes.
- `market.risk`: a marketRisk S3 object.
- `with.constant`: a logical value. Should the expression be with constant or not?

**Value**

a function representing the aggregated valuation function.
**keywordToTable**

*Extract a table from the excel template*

**Description**

This function extracts tables from the excel input workbook.

**Usage**

```
keywordToTable(path, keyword, mapping.tables, keep = NULL, colNames = NULL)
```

**Arguments**

- `path`: a character value. A valid path of an input excel workbook. The path can be relative or not.
- `keyword`: a character value. A valid keyword corresponding to a table.
- `mapping.tables`: a data.frame with following fields:
  - `keyword`: character, list of keywords.
  - `name`: character, sheet names corresponding to the keywords.
  - `startRow`: integer, starting row number corresponding to position in the excel sheet.
  - `startCol`: integer, starting column number corresponding to position in the excel sheet.
  - `endCol`: integer, ending column number corresponding to position in the excel sheet.
- `keep`: integer vector, which columns should be kept or removed from `startCol:endCol`.
- `colNames`: character vector, the colnames to be given to the parsed table.

**Value**

The corresponding table.

**See Also**

`excelToSstModel`
### keywordToTransposedTable

*Extract a table from the excel template*

**Description**

This function extracts transposed tables from the excel input workbook.

**Usage**

```r
keywordToTransposedTable(path, keyword, mapping.Ntables, colNames = NULL)
```

**Arguments**

- **path**
  - A character value. A valid path of an input excel workbook. The path can be relative or not.

- **keyword**
  - A character value. A valid keyword corresponding to a table.

- **mapping.tables**
  - A data.frame with following fields:
    - **keyword**: character, list of keywords.
    - **name**: character, sheet names corresponding to the keywords.
    - **startrow**: integer, starting row number corresponding to position in the excel sheet.
    - **startcol**: integer, starting column number corresponding to position in the excel sheet.

- **colNames**
  - A character vector, the colnames to be given to the parsed table.

**Value**

The corresponding table.

**See Also**

`excelToSstModel`

### keywordToValue

*Extract a value from the excel template*

**Description**

This function extracts single values from the excel input workbook.

**Usage**

```r
keywordToValue(path, keyword, mapping.values)
```
launchDashboard

Arguments

path a character value. A valid path of an input excel workbook. The path can be relative or not.

keyword a character value. A valid keyword corresponding to a cell.

mapping.values a data.frame with three columns:
  • keyword: character, list of keywords.
  • name: character, sheet names corresponding to the keywords.
  • row: integer, row number corresponding to position in the excel sheet.
  • col: integer, column number corresponding to position in the excel sheet.

Value

the value of the corresponding cell.

See Also

toExcelModel.

Description

This function launch an interactive dashboard for SST computations.

Usage

launchDashboard()

Value

None (intended for side-effects)
liability

Constructing an Insurance Liability

Description

Constructor for the S3 class liability. It allows to build for an insurance liability referred under the name "Versicherungsverpflichtungen" in the FINMA technical document "SST-Marktrisiko und -Aggregation Technische Beschreibung".

Usage

liability(time, currency, value)

Arguments

time strictly positive integer value of length one representing the time-to-maturity. This parameter relates to the "Restlaufzeit" liability variable tau in the FINMA document "SST-Marktrisiko und -Aggregation Technische Beschreibung".

currency character value of length one representing the currency in which the fixed-income-asset is labeled. This parameter relates to the "Fremdwährungsrisiko-faktor" index j in the FINMA document "SST-Marktrisiko und -Aggregation Technische Beschreibung".

value non-zero numeric value of length one representing the "Certainty-Equivalent-Versicherungsverpflichtung-Cashflows" as referred in the FINMA document "SST-Marktrisiko und -Aggregation Technische Beschreibung". at time time. This must be expressed in the same currency as currency. If value is negative, then the liability is interpreted as a positive cashflow.

Value

an S3 object, instance of the class liability.

See Also

summary.liability.print.liability.

Examples

# Creating new liabilities.
liability1 <- liability(1, "USD", 1000)
liability2 <- liability(2, "EUR", 2000)
Constructing a Life Delta-Normal Remainder Term with Respect to lifeRisk

Description

Constructor for the S3 class life. It allows to build for the sensitivities with respect to the life risk factors of the total positions not modeled by the other marketItems.

Usage

life(name, currency, sensitivity)

Arguments

name character value. The names of the life risk-factors (the life risk factors defined in lifeRisk) with respect to which sensitivities are computed (non-zero). This vector should not contain duplicated names.

currency character value representing currencies in which the sensitivities are expressed. If the currency specified does not match the base currency of the marketRisk, the initial fx-rates will be used to convert to the base currency. Nevertheless, it is forced at construction of a portfolio that the sensitivities should be provided in the portfolio base currency.

sensitivity numeric value giving the sensitivities (understood as quantiles) for the corresponding life risk-factors provided in name. Please consult the help page of lifeRisk for more information on the meaning of these sensitivities. Sensitivities must be expressed in the corresponding currencies in currency.

Value

an S3 object, instance of the class life.

Note

All parameters must be of equal length.

See Also

summary.life, print.life.

Examples

# Creating a new health.
life1 <- life(name = c("pandemy", "lontevity", "storno"),
currency = c("EUR", "CHF", "EUR"),
sensitivity = c(100, 150, 130))
Constructing a LifeRisk

Description

lifeRisk is the constructor for the S3 class lifeRisk. It allows to build for life insurance risks parameters.

Usage

lifeRisk(corr.mat, quantile)

Arguments

corr.mat  
matrix of numeric values. This must be a valid correlation matrix and should have names, i.e. attributes colnames and rownames indicating the names of the corresponding life insurance risk-factors.

quantile  
positive numeric value smaller than one representing the probabilities at which the life sensitivities will be interpreted as (1-quantile)-quantiles.

Value

an S3 object, instance of the class lifeRisk.

See Also


Examples

# Creating new lifeRisks.
corr.mat <- diag(rep(1, 2))
colnames(corr.mat) <- c("invalidity", "longevity")
rownames(corr.mat) <- colnames(corr.mat)

lifeRisk1 <- lifeRisk(corr.mat = corr.mat,
                        quantile = c(0.995, 0.995))
**logNormalExpression**  
*Log-Normal Expression Helper*

**Description**
This private function creates a log-normal expression.

**Usage**

```r
logNormalExpression(object, market.risk, standalone)
```

**Arguments**
- `object`  
a S3 object of class `item`.
- `market.risk`  
a S3 object of class `marketRisk`.
- `standalone`  
a S3 object of class `standalone`.

**Value**
a character value.

---

**macroEconomicScenarios**  
*Constructing Macro Economic Scenarios*

**Description**
`macroEconomicScenario` is an S3 method to construct macro economic scenarios, i.e. constrained values taken by the market risk-factors and potentially participation.

**Usage**

```r
macroEconomicScenarios(macro.economic.scenario.table)
```

**Arguments**
- `macro.economic.scenario.table`  
a numeric matrix with named columns and rows. Each row represents a different economic scenario and each column is associated to a risk-factor appearing in a `marketRisk` (or a participation contained in the portfolio). The rownames of the matrix should indicate the names of the economic scenarios and the columns names should exactly match the names of the base risk-factors defined in a `marketRisk` (respecting the order). In addition, if the underlying portfolio also contains a participation, an additional column named "participation" should be included in this table as the last column.
mappingTable

Description

mappingTable is the constructor for the S3 class mappingTable. It allows to define the market risk factors.

Usage

mappingTable(..., list.arg = F)

Arguments

... riskFactor objects. Please note that no risk factor name can be chosen among the following reserved words (in that case it would trigger an error):

- marketRisk
- lifeRisk
- healthRisk
- nonLifeRisk
- scenarioRisk
- participationRisk
- participation
- marketParticipationRisk
- asset
- cashflow
- liability
- assetForward
- fxForward
- delta

list.arg a logical value, by default set to FALSE. It allows to use ... argument to pass a list of objects of class riskFactor.

Value

An S3 object, instance of the class mappingTable.
Constructing Time Mappings

**Description**
Constructing Time Mappings

**Usage**
mappingTime(time, mapping)

**Arguments**
time integer value, the time to maturities.
mapping character value, the mapping.

**Value**
a data.frame with option stringsAsFactors = FALSE.

**See Also**
marketRisk.

Constructing a MarketRisk

**Description**
marketRisk is the constructor for the S3 class marketRisk. It allows to build for market risk parameters.

**Usage**
marketRisk(cov.mat, mapping.table, initial.values, mapping.time, base.currency)

**Arguments**
cov.mat numeric matrix. The covariance matrix of the market risk-factors. This matrix must have names, i.e. attributes colnames and rownames indicating the names of the corresponding market risk-factors, please note that "participation" is a reserved name and should not be used. This matrix should also have an attribute named "base.currency" indicating to which currency the fx rates are mapped in the covariance matrix (use the function attr()).
mapping.table S3 object created using the constructor mappingTable.
initial.values list with the following elements:

- initial.fx: a data.frame with following columns and parameters:
  - from: a character value. The starting currencies.
  - to: a character value. The arrival currencies.
  - fx: a numeric value. The exchange rates from the starting currencies to the arrival currencies.

- initial.rate: a data.frame with following columns and parameters:
  - time: an integer value. The terms for the interests.
  - currency: a character value. The currencies for the interest rates.
  - rate: a numeric value. The interest rates.

Please note that you can directly use the constructors initialFX and initialRate to provide these parameters.

mapping.time a data.frame with following columns and parameters:

- time-to-maturity: an integer value. The times to maturities.
- mapping: character value. The mapping.
- stringsAsFactors = FALSE.

Please note that you can directly use the constructor mappingTime to provide this parameter.

base.currency a character value of length one, the base currency of the marketRisk.

Value

S3 object, instance of the class marketRisk.

See Also

mappingTable.

---

marketValueMargin  
*Compute the Market Value Margin (MVM)*

Description

S3 generic method to compute the market value margin.

Usage

marketValueMargin(object, ...)

Arguments

object an S3 object.

... additional parameters.

Value

a numeric value.
marketValueMargin.sstOutput

Compute the Market Value Margin (MVM)

Description

S3 generic method to compute the market value margin (MVM).

Usage

```r
## S3 method for class 'sstOutput'
marketValueMargin(object, nhmr = NULL, ...)
```

Arguments

- `object`  
  S3 object of class sstOutput.

- `nhmr`  
  numeric value of length one. The factor for non-headgeable market risk in market value margin computations. Default to NULL, in this case the sstOutput must contain this parameter. This parameter overrides nhmr in objects of class sstOutput.

- `...`  
  aditional parameters to be passed on to expectedShortfall.

Value

- a numeric value of length one. The market value margin (MVM).

See Also

- `marketValueMargin`

mvmLife  

MVM life computation

Description

compute MVM life.

Usage

```r
mvmLife(cashflow.table, rates, cov.mat, coc)
```
Arguments

cashflow.table  a data.table.

rates           a numeric vector of rates, with continuous compounding. These should start for
time to maturity 1 and go until 1 + last cashflow time to maturity.

cov.mat         covariance matrix of life risks.

coc             a numeric value. The cost of capital.

Value

a numeric value, the life MVM.

na.rm           Remove Missing Values

Description

na.rm removes all missing values from a vector.

Usage

na.rm(x)

Arguments

x               an atomic vector.

Value

an atomic vector without NA values.

newtonRaphson   Find roots using Newton-Raphson algorithm

Description

find root of a function using the Newton Raphson algorithm.

Usage

newtonRaphson(f, df, start = 0, atol = 1e-04, rtol = 1e-04,
               maxit = 10000, ...)
nonLifeRisk

Arguments

- **f**: a numeric valued function from a single numeric argument.
- **df**: the derivative of ‘f’.
- **start**: numerical value. The initial position for the Newton Raphson iteration.
- **atol**: numeric value. The absolute tolerance for finding a root.
- **rtol**: numeric value. The relative tolerance for finding a root.
- **maxit**: an integer value. The maximal number of iterations.
- **...**: additional parameters to be passed to ‘f’.

Value

- a numeric value, the root.

---

**nonLifeRisk**

*Constructing a nonLifeRisk*

Description

nonLifeRisk is the constructor for the S3 class nonLifeRisk. It allows to build for non-life insurance risks simulations.

Usage

nonLifeRisk(type, param, currency)

Arguments

- **type**: a character value of length one indicating the type of simulation used. It can be one of the following option:
  - "simulations": simulations for non-life risk are directly provided.
  - "log-normal": simulations for non-life risk are assumed to come from log-normal random variables.
  - "cdf": simulations from non-life risk are simulated from an input cumulative distribution function.
- **param**: a list of length one or two depending on the type chosen. The structure of the list is conditional on the type of nonLifeRisk:
  - if `type = "simulations"`, then `param` should be a named list with one element:
    - `simulations`: numeric value representing the input simulations. If the number of input simulations are bigger or equal to the number of required simulations, then inputs are subsampled. In the other, bootstrap is used.
  - if `type = "log-normal"`, then `param` should be a named list with two elements:
    - `mu`: numeric value of length one giving the drift of the log-normal variable.
• sigma: strictly positive numeric value of length one giving the volatility of the log-normal variable.

if type = "cdf" then param should be a named list with one element:

• cdf: a data.frame with two columns named x and cdf, where the column x contains the numeric values that represents the possible discrete values of the CDF and cdf the cumulative distribution function evaluated at these possible values. Please note that we require the user to provide both columns in an increasing order. We additionally require the user to provide a value for cdf == 1 in order to know all the jumps possibly taken, since the cdf is right-continuous. Moreover please consider that we interpret the CDF as a piece-wise right-continuous step function.

currency a character value. representing the currency in which the simulations are expressed. Please note that currency is restricted to be the same as the marketRisk used in conjunction.

Value

an S3 object, instance of the class nonLifeRisk.

Note

In case of log-normal simulations, a warning is triggered if the parameters seem to be not reasonable and could eventually yield non-finite simulations.

See Also

summary.nonLifeRisk, print.nonLifeRisk, simulate.nonLifeRisk, compute.nonLifeRisk.

Examples

# Creating new nonLifeRisks
nonLife1 <- nonLifeRisk(type = "simulations",
    param = list(simulations = stats::rnorm(100)),
    currency = "CHF")
nonLife2 <- nonLifeRisk(type = "log-normal",
    param = list(mu = 1, sigma = 2),
    currency = "CHF")
nonLife3 <- nonLifeRisk(type = "cdf",
    param = list(cdf = data.frame(x = c(0,1,2,3),
                              cdf = c(0.3,0.7,0.9, 1))),
    currency = "CHF")
Description

`participation` is the constructor for the S3 class `participation`. It allows to build for a participation position.

Usage

`participation(currency, value)`

Arguments

- **currency**: character value of length one. The currency in which the participation is expressed.
- **value**: positive numeric value of length one. The total value of the participation. This must be expressed in the same currency as `currency`.

Value

an S3 object, instance of the class `participation`.

Note

Please note that combined with a portfolio, the participation should be provided in the base currency.

See Also

`summary.participation`, `print.participation`.

Examples

```r
# Creating new participations.
participation1 <- participation("USD", 1000)
participation2 <- participation("EUR", 2000)
```
**participationRisk**  
*Constructing a participationRisk*

**Description**

participationRisk is the constructor for the S3 class participationRisk. It allows to build for participation risk parameters.

**Usage**

participationRisk(volatility)

**Arguments**

volatility positive numeric value of length one.

**Value**

An S3 object, instance of the class participationRisk.

**See Also**

summary.participationRisk, print.participationRisk, compute.participationRisk.

**Examples**

# Creating a new participationRisk.
pr <- participationRisk(volatility = 0.5)

---

**pcRate**  
*Constructing a Principal Component Rate (Risk Factor)*

**Description**

Constructor for the S3 class pcRate. It allows to define a principal component of rate curves risk factor. This risk factor refers to a principal component in the decomposition of the "stetigen Zins" change $\Delta R_{ij}(t, i, \tau)$ for a certain horizon index $i$, and a certain currency $j$ in the valuation function for "Fixed-Income-Assets und Versicherungsverpflichtungen" presented in the FINMA document "SST-Marktrisiko und -Aggregation Technische Beschreibung".

**Usage**

pcRate(name, currency, scale = NULL)
**portfolio**

**Arguments**

- **name**
  a character value of length one. This corresponds to the name in the covariance matrix of the `marketRisk` to which the principal component rate risk factor is mapped. This means that the principal component change will be assumed to be modeled by the underlying normal random variable corresponding to `name` in the covariance matrix (potentially scaled by `scale` if not `NULL`).

- **currency**
  a character value of length one. The currency in which the underlying rate is modelling. This refers to the currency corresponding to the index \( j \) in the FINMA document "SST-Marktrisiko und -Aggregation Technische Beschreibung".

- **scale**
  a numeric value of length one. If not set `NULL`, this defines a scaled risk factor equal to `scale` times the risk factor defined by `name` in the covariance matrix contained in `marketRisk`. By default its value is `scale = NULL`.

**Value**

An S3 object, instance of the class `pcRate`.

**Examples**

```r
# constructing a principal component rate risk factor  
# (assuming "2Y_CHF" exists in marketRisk).  
p <- pcRate(name = "pcRate_EUR_1", currency = "EUR")
```

---

**portfolio**  
*Constructing a SST Portfolio*

**Description**

Constructor for the S3 class `portfolio`. It allows to build for a sst portfolio containing financial items (market items), insurance items (life and health) as well as a participation.

**Usage**

```r
portfolio(market.items = NULL, participation.item = NULL,  
  life.item = NULL, health.item = NULL, base.currency, portfolio.parameters)
```

**Arguments**

- **market.items**
  a list of `marketItem` S3 objects created using the constructors (see the corresponding help pages for more information):
    - `asset`
    - `cashflow`
    - `liability`
    - `assetForward`
    - `fxForward`
• delta

Please refer to the note Section for extra-information.

**participation**

a participation S3 object created using the constructor participation. This should be expressed in the same currency as base.currency.

**life**

a life S3 object created using the constructor life. The life sensitivities be expressed in the same currency as base.currency.

**health**

a health S3 object created using the constructor health. The health sensitivities be expressed in the same currency as base.currency.

**base.currency**

a character value representing the base currency in which the holder of the portfolio reports its results.

**portfolio.parameters**

a list of parameters specific to the portfolio (understood in currency base.currency) with entries:

- **mvm**: market value margin information (MVM), this should be a named list with three numeric fields of length one:
  - **mvm.life**: the market value margin for life;
  - **mvm.health**: the market value margin for health;
  - **mvm.nonlife**: the market value margin for non-life.

- **rtkr**: risk-bearing capital (RBC) at time 0 run-off, this should be a numeric value of length one.

- **rtkg**: risk-bearing capital (RBC) at time 0 on-going concern, this should be a numeric value of length one.

- **credit.risk** the credit risk value, this should be a numeric value of length one.

- **expected.insurance.result** expected insurance result, this should be a numeric value of length one.

- **expected.financial.result** expected financial result, this should be a numeric value of length one.

- **correction.term** correction term, this should be a numeric value of length one.

**Note**

In order to create an sstModel, the portfolio should contain at least one marketItem. Additionally, we do not allow for a portfolio containing a participation without any marketItem.

**See Also**

summary.portfolio, print.portfolio, asset, cashflow, liability, fxForward, assetForward, delta, participation, life, nonLifeRisk, health, scenarioRisk.

**Examples**

# Creating a portfolio.
asset1 <- asset("equity", "USD", 1000)
print.asset

Printing an Asset with Direct Market Price

Description

print method for the S3 class asset.

Usage

## S3 method for class 'asset'
print(x, ...)

Arguments

x S3 object of class asset.

... additional arguments.

Value

None (invisible NULL).

See Also

print.asset

asset2 <- asset("hedge fund", "EUR", 2000)
life1 <- life(name = c("pandemy", "longevity", "storno"),
currency = c("CHF", "CHF", "CHF"),
sensitivity = c(-100, -150, -130))
health1 <- health(name = c("pandemy", "longevity", "storno"),
currency = c("CHF", "CHF", "CHF"),
sensitivity = c(100, 150, 130))
participation1 <- participation("CHF", 1000)
# valid portfolio parameters
valid.param <- list(mvm = list(mvm.life = 2, mvm.health = 4, mvm.nonlife = 3),
rtkr = 0,
rtkg = 0,
correction.term = 2,
credit.risk = 3,
expected.insurance.result = 10^6,
expected.financial.result = 10^5)
 pf <- portfolio(market.items = list(asset1, asset2),
 participation.item = participation1,
 life.item = life1,
 health.item = health1,
 base.currency = "CHF",
 portfolio.parameters = valid.param)
print.assetForward

**Examples**

```r
# Creating an asset.
a <- asset("equity", "USD", 1000)
# printing the asset.
print(a)
```

**Description**

print method for the S3 class assetForward.

**Usage**

```r
## S3 method for class 'assetForward'
print(x, ...)
```

**Arguments**

- `x` S3 object of class assetForward.
- `...` additional arguments.

**Value**

None (invisible NULL).

**See Also**

- `print.assetForward`

**Examples**

```r
# Creating an assetForward.
af <- assetForward("equity", "EUR", 1, 1000, 1200, "long")
# printing the assetForward.
print(af)
```
print.cashflow

Printing a Fixed-Income-Asset

Description
print method for the S3 class cashflow.

Usage
## S3 method for class 'cashflow'
print(x, ...)

Arguments
x S3 object of class cashflow.
...
additional arguments.

Value
None (invisible NULL).

See Also
print, cashflow

Examples
# Creating a cashflow.
ca <- cashflow(1L, "USD", "AAA", 0.5, 1000)
# printing the cashflow.
print(ca)

print.delta

Printing a Delta-Normal Remainder Term

Description
print method for S3 class delta.

Usage
## S3 method for class 'delta'
print(x, ...)

Arguments

x an S3 object of class delta.

Value

None (invisible NULL).

See Also

print, delta.

Examples

# Creating a new delta.
delta1 <- delta('name' = c("equity", "2YCHF", "EURCHF"),
currency = c("EUR", "CHF", "EUR"),
sensitivity = c(100, 150, 130))

# printing the delta.
print(delta1)
Examples

# Creating an fx forward.
ffx <- fxForward("USD", "EUR", 1, 1000, 1.05, "long")
# printing the fx forward.
print(ffx)

print.health    Printing a Health Delta-Normal Term

Description

print method for the S3 class health.

Usage

## S3 method for class 'health'
print(x, ...)

Arguments

x          S3 object of class health.
...        additional arguments.

Value

None (invisible NULL).

See Also

print, health.

Examples

# Creating a new health item.
health1 <- health(name = c("pandemy", "longetivity", "storno"),
                  currency = c("EUR", "CHF", "EUR"),
                  sensitivity = c(100, 150, 130))
# printing the health item.
health1
Description

print method for the S3 class healthRisk.

Usage

```r
## S3 method for class 'healthRisk'
print(x, ...)
```

Arguments

- `x`: S3 object of class healthRisk.
- `...`: additional arguments.

Value

None (invisible NULL).

See Also

`print`, `healthrisk`.

Examples

```r
# Creating a new healthRisk.
corr.mat <- diag(rep(1, 2))
colnames(corr.mat) <- c("invalidity", "longetivity")
rownames(corr.mat) <- colnames(corr.mat)

healthRisk1 <- healthRisk(corr.mat = corr.mat)
# printing the healthRisk.
print(healthRisk1)
```
print.liability

**Description**

print method for the S3 class `liability`.

**Usage**

```r
## S3 method for class 'liability'
print(x, ...)
```

**Arguments**

- `x`: an S3 object of class `liability`.
- `...`: additional arguments.

**Value**

None (invisible NULL).

**See Also**

`print`, `liability`.

**Examples**

```r
# Creating a liability.
liab <- liability(1, "USD", 1000)
# printing the liability.
print(liab)
```

print.life

**Description**

print method for S3 class `life`.

**Usage**

```r
## S3 method for class 'life'
print(x, ...)
```
Arguments

x  an S3 object of class lifeRisk.

... additional parameters.

Value

None (invisible NULL).

See Also

print, life.

Examples

```r
# Creating a new health item.
life1 <- life(name = c("pandemy", "longevity", "storno"),
              currency = c("EUR", "CHF", "EUR"),
              sensitivity = c(100, 150, 130))
# printing the health item.
life1
```

---

**Description**

print method for the S3 class lifeRisk.

**Usage**

```r
## S3 method for class 'lifeRisk'
print(x, ...)
```

**Arguments**

x  an S3 object of class lifeRisk.

... additional arguments.

**Value**

None (invisible NULL).

**See Also**

print, lifeRisk.
Examples

# Creating a new lifeRisk.

corr.mat <- diag(rep(1, 2))
colnames(corr.mat) <- c("invalidity", "longevity")
rownames(corr.mat) <- colnames(corr.mat)

lifeRisk1 <- lifeRisk(corr.mat = corr.mat,
    quantile = c(0.995, 0.995))

# printing the lifeRisk.
print(lifeRisk1)

---

print.marketRisk  Printing a marketRisk

Description

print method for the S3 class marketRisk.

Usage

## S3 method for class 'marketRisk'
print(x, ...)

Arguments

x  S3 object of class marketRisk.
...  additional arguments.

Value

None (invisible NULL).

See Also

print.marketRisk.
print.nonLifeRisk  

**Printing a nonLifeRisk**

**Description**

print method for the S3 class nonLifeRisk.

**Usage**

```r
t## S3 method for class 'nonLifeRisk'
print(x, ...)
```

**Arguments**

- `x`:
  - an S3 object of class nonLifeRisk.

- `...`:
  - additional arguments.

**Value**

None (invisible NULL).

**See Also**

`print`, `nonliferisk`.

**Examples**

```r
# Creating a new nonLifeRisk.
nonlife1 <- nonLifeRisk(type = "simulations",
                        param = list(simulations = stats::rnorm(100)),
                        currency = "CHF")
# printing the nonLifeRisk.
print(nonlife1)
# Creating a new nonLifeRisk.
nonlife2 <- nonLifeRisk(type = "log-normal",
                        param = list(mu = 1, sigma = 2),
                        currency = "CHF")
# printing the nonLifeRisk.
print(nonlife2)
# Creating a new nonLifeRisk.
nonlife3 <- nonLifeRisk(type = "cdf",
                        param = list(cdf = data.frame(x = c(0,1,2,3),
                                                  cdf = c(0.3,0.7,0.9, 1))),
                        currency = "CHF")
# printing the nonLifeRisk.
print(nonlife3)
```
print.participation  Printing a Participation

Description
print method for the S3 class participation.

Usage

```r
## S3 method for class 'participation'
print(x, ...)
```

Arguments

- `x` S3 object of class participation.
- `...` additional arguments.

Value
None (invisible NULL).

See Also

- `print`, `participation`.

Examples

```r
# Creating a new participation.
participation1 <- participation("USD", 1000)
# printing the participation
participation1
```

print.participationRisk  Printing a participationRisk

Description
print method for S3 class participationRisk.

Usage

```r
## S3 method for class 'participationRisk'
print(x, ...)
```
print.portfolio

Arguments

x an S3 object of class participationRisk.
... additional parameters.

Value

None (invisible NULL).

See Also

print, participationRisk.

Examples

# Creating a new participationRisk.
pr <- participationRisk(volatility = 0.5)
# printing the participationRisk.
pr

print.portfolio Printing a Portfolio

Description

print method for the S3 class portfolio.

Usage

## S3 method for class 'portfolio'
print(x, ...)

Arguments

x S3 object of class portfolio.
... additional arguments.

Value

None (invisible NULL).

See Also

print, participation.
Examples

# Creating a new portfolio.
asset1 <- asset("equity", "USD", 1000)
asset2 <- asset("hedge fund", "EUR", 2000)
life1 <- life(name = c("pandemy", "longevity", "storno"),
              currency = c("CHF", "CHF", "CHF"),
              sensitivity = c(100, 150, 130))
health1 <- health(name = c("pandemy", "longevity", "storno"),
                   currency = c("CHF", "CHF", "CHF"),
                   sensitivity = c(100, 150, 130))
participation1 <- participation("CHF", 1000)
valid.param <- list(mvm = list(mvm.life = 2, mvm.health = 4, mvm.nonlife = 3),
                     rtkr = 0,
                     rtkg = 0,
                     correction.term = 2,
                     credit.risk = 3,
                     expected.insurance.result = 10^6,
                     expected.financial.result = 10^5)
pf <- portfolio(market.items = list(asset1, asset2),
                participation.item = participation1,
                life.item = life1,
                health.item = health1,
                base.currency = "CHF",
                portfolio.parameters = valid.param)

# printing the portfolio
print(pf)

print.scenarioRisk  

Printing a ScenarioRisk

Description

print method for the S3 class scenarioRisk.

Usage

## S3 method for class 'scenarioRisk'
print(x, ...)

Arguments

  x            S3 object of class scenarioRisk.

  ...          additional arguments.

Value

  None (invisible NULL).
See Also

print, scenarioRisk.

Examples

# Creating a new scenarioRisk.
scenarios <- scenarioRisk(name = c("earthquake", "real estate crash"),
                          probability = c(0.001, 0.01),
                          currency = c("CHF", "CHF"),
                          effect = c(-1000, -10000))

# printing the scenarioRisk.
print(scenarios)

print.sstModel  Printing a sstModel

Description

print method for the S3 class sstModel.

Usage

## S3 method for class 'sstModel'
print(x, ...)

Arguments

x  S3 object of class sstModel.
...
additional arguments.

Value

None (invisible NULL).

See Also

print.sstModel.
print.sstOutput

Printing a sstOutput

Description
print method for S3 class sstOutput.

Usage

```r
## S3 method for class 'sstOutput'
print(x, ...)
```

Arguments

- `x` S3 object of class sstOutput.
- `...` additional arguments.

Value
None (invisible NULL).

See Also
print.

print.standalone

Printing a standalone

Description
print method for the S3 class standalone.

Usage

```r
## S3 method for class 'standalone'
print(x, ...)
```

Arguments

- `x` S3 object of class standalone.
- `...` additional arguments.

Value
None (invisible NULL).
print.summary.portfolio

Printing a Summary of Portfolio

Description

print method for S3 class summary.portfolio.

Usage

## S3 method for class 'summary.portfolio'
print(x, ...

Arguments

x

an S3 object of class summary.portfolio.

... additional parameters.

Value

None (invisible NULL).

See Also

print


print.summary.sstModel

Printing a Summary of sstModel

Description

print method for S3 class summary.sstModel.

Usage

## S3 method for class 'summary.sstModel'
print(x, ...

Arguments

x

an S3 object of class summary.sstModel.

... additional parameters.
Value
None (invisible NULL).

See Also
print

Description
print method for S3 class summary.sstOutput.

Usage
## S3 method for class 'summary.sstOutput'
print(x, ...)

Arguments
x S3 object of class summary.sstOutput.
... additional arguments.

Value
None (invisible NULL).

See Also
print.

rate Constructing a Rate (Risk Factor)

Description
Constructor for the S3 class rate. It allows to define a rate-type risk factor. This risk factor refers to the "stetigen Zins" change $\Delta R_j(t, i)$ for a certain horizon index $i$, and a certain currency $j$ in the valuation function for "Fixed-Income-Assets und Versicherungsverpflichtungen" presented in the FINMA document "SST-Marktrisiko und -Aggregation Technische Beschreibung".

Usage
rate(name, currency, horizon, scale = NULL)
Arguments

name
a character value. If the length is one, this corresponds to the name in the covariance matrix of the marketRisk to which the rate risk factor is mapped. This means that the risk factor change $\Delta R_j(t, \tau)$ in the FINMA document "SST-Marktrisiko und -Aggregation Technische Beschreibung" (version 31.1.2018) will be assumed to be modeled by the underlying normal random variable corresponding to name in the covariance matrix (potentially scaled by scale if not NULL). If the length is strictly greater than one, this corresponds to multiple names in the covariance matrix of the marketRisk to which the rate risk factor is mapped in the case of principal component modeling. This means that the risk factor change $\Delta R_j(t, \tau)$ in the FINMA document "SST-Marktrisiko und -Aggregation Technische Beschreibung" will be assumed to be modeled by a linear combination (with coefficients scale) of normal random variable corresponding to the multiple names name in the covariance matrix. Please refer to the note section to have more information.

currency
a character value of length one. The currency in which the underlying "Fixed-Income-Assets oder Versicherungsverpflichtungen" is valuated. This refers to the currency corresponding to the index $j$ in the FINMA document "SST-Marktrisiko und -Aggregation Technische Beschreibung".

horizon
a character value of length one. The time-to-maturity (projected on the time mapping). This refers to the index $i$, in the FINMA document "SST-Marktrisiko und -Aggregation Technische Beschreibung".

scale
a numeric value of length one. If not set NULL, this defines a scaled risk factor equal to scale times the risk factor defined by name in the covariance matrix contained in marketRisk. By default its value is scale = NULL. In the case of principal component modeling (i.e. name of length strictly greater than one) this parameter should be provided as a numeric values of the same length as name corresponding to the loadings in the principal component decomposition. Please consider that these loadings should be contained in the Euclidean disk, i.e. the sum of there squared value should be below 1, if not a warning will be triggered.

Value

An S3 object, instance of the class rate.

Note

In the case that principal component modeling of rate curves is chosen, all risk factors named in name should be scaled, otherwise an error will be triggered.

Examples

# constructing a non-scaled rate risk factor
# (assuming "2Y_CHF" exists in marketRisk).

r <- rate(name = "2Y_CHF",
currency = "CHF",
horizon = "k")
## rateIsIn

**rateIsIn**

### Rate in Object?

#### Description
S3 generic to check that the rate is in the object.

#### Usage

```r
rateIsIn(object, ...)
```

#### Arguments

- `object`: an S3 object potentially containing the rate.
- `...`: additional parameters.

#### Value

a logical value.

---

## rateIsIn.standalone

**rateIsIn.standalone**

### Rate in standalone?

#### Description
S3 generic to check that the rate is in the object.

#### Usage

```r
## S3 method for class 'standalone'
rateIsIn(object, currency, horizon, ...)
```
removePerfectCorr

Arguments

object S3 object of class standalone.
currency character value. A currency.
horizon character value. An horizon.
... additional arguments.

Value

a logical value, is the rate in the standalone?

See Also

rateIsIn.

removePerfectCorr Remove Perfectly Correlated Variables

Description

remove perfectly correlated variables from a matrix

Usage

removePerfectCorr(mat)

Arguments

mat a numeric matrix

Value

a sub matrix
riskCapital  

Compute the Risk Capital

Description
S3 generic method to compute the risk capital.

Usage
riskCapital(object, ...)

Arguments
- object: an S3 object.
- ...: additional parameters.

Value
a numeric value.

riskCapital.sstOutput  

Compute the Risk Capital (RC)

Description
S3 generic method to compute the risk capital (RC).

Usage
```r
## S3 method for class 'sstOutput'
riskCapital(object, with.scenario = F, ...)
```

Arguments
- object: S3 object of class sstOutput.
- with.scenario: logical value of length one. Should the risk capital be compute with scenario risk also?
- ...: additional parameters to be passed on to expectedShortfall.

Value
a numeric value. The risk capital (RC).

See Also
riskCapital.
riskFactorToExpression

*RiskFactor To Expression Helper*

**Description**

This private function creates an expression from a risk-factor.

**Usage**

```r
riskFactorToExpression(risk.factor)
```

**Arguments**

- `risk.factor`: a riskFactor object.

**Value**

a character value.

---

scenarioRisk

*Constructing a scenarioRisk*

**Description**

scenarioRisk is the constructor for the S3 class scenarioRisk. It allows to build for scenarios (stress-tests).

**Usage**

```r
scenarioRisk(name, probability, currency, effect)
```

**Arguments**

- `name`: character value. The names of the scenarios. This should not contain duplicated names.
- `probability`: numeric value. The probability of the respective scenarios. Probabilities must take values between 0 and 1, i.e. must be in (0, 1).
- `currency`: character value. The currencies in which the effect are expressed. Please note that currency is restricted to be the same as the base currency of a marketRisk.
- `effect`: numeric value. The effects associated with each scenario on the risk-bearing-capital (RBC). This must be expressed in the same currency as `currency`.  

---
**simulate.healthRisk**

Value

An S3 object, instance of the class `scenarioRisk`.

Note

All parameters must be of equal length.

See Also

`summary.scenarioRisk`, `print.scenarioRisk`, `simulate.scenarioRisk`, `compute.scenarioRisk`.

Examples

```r
C creating new scenarioRisk.
scenarios <- scenarioRisk(name = c("earthquake", "real estate crash"),
                          probability = c(0.001, 0.01),
                          currency = c("CHF", "CHF"),
                          effect = c(1000, 10000))
```

---

**Simulate from a HealthRisk**

Description

`simulate` is a generic S3 method for classes inheriting from `risk`. It returns a vector of risk-factor simulations for the corresponding risk.

Usage

```r
# S3 method for class 'healthRisk'
simulate(object, nsim, seed = NULL, ...)
```

Arguments

- `object` S3 object of class `healthRisk`.
- `nsim` strictly positive integer value of length one. The number of simulations.
- `seed` positive integer value of length one. The seed for reproducibility.
- `...` additional argument to be passed to `rnorm`.

Value

A numeric value, the base simulations.

See Also

`simulate`, `healthRisk`.  

**simulate.lifeRisk**  
*Simulate from a LifeRisk*

Description

`simulate` is a generic S3 method for classes inheriting from `risk`. It returns a vector of risk-factor simulations for the corresponding risk.

Usage

```r
## S3 method for class 'lifeRisk'
simulate(object, nsim, seed = NULL, ...)
```

Arguments

- `object`  
  S3 object of class `lifeRisk`
- `nsim`  
  strictly positive integer value of length one. The number of simulations.
- `seed`  
  positive integer value of length one. The seed for reproducibility.
- `...`  
  additional arguments to be passed to `rnorm`.

Value

a numeric value, the base simulations.

See Also

`simulate.lifeRisk`.

**simulate.marketRisk**  
*Simulate from a MarketRisk*

Description

`simulate` is a generic S3 method for classes inheriting from `risk`. It returns a vector of risk-factor simulations for the corresponding risk.

Usage

```r
## S3 method for class 'marketRisk'
simulate(object, nsim, seed = NULL, DT = FALSE, ...)
```
Arguments

object: object of class marketRisk.
nsim: strictly positive integer value of length one. The number of simulations.
seed: positive integer value of length one. The seed for reproducibility.
DT: a boolean value, should we cast the simulation matrix in a data.table?
...: additional arguments.

Value

a matrix or data.table of base simulations.

See Also

simulate, marketRisk.
simulate.participationRisk

Simulate from a participationRisk

Description

simulate is a generic S3 method for classes inheriting from risk. It returns a vector of risk-factor simulations for the corresponding risk.

Usage

```r
## S3 method for class 'participationRisk'
simulate(object, nsim, seed = NULL, ...)
```

Arguments

- `object`: S3 object of class participationRisk.
- `nsim`: strictly positive integer value og length one. The number of simulations.
- `seed`: positive integer value of length one. The seed for reproducibility.
- `...`: additional arguments.

Value

a numeric value. The base simulations.

See Also

`simulate`, `participationRisk`, `participation`.

simulate.scenarioRisk

Simulate from a ScenarioRisk

Description

simulate is a generic S3 method for classes inheriting from risk. It returns a vector of risk-factor simulations for the corresponding risk.

Usage

```r
## S3 method for class 'scenarioRisk'
simulate(object, nsim, seed = NULL, market.risk, ...)
```
Arguments

object  S3 object of class scenarioRisk.
nsim    strictly positive integer value of length one. The number of simulations.
seed    positive integer value of length one. The seed for reproducibility.
market.risk  S3 object of class marketRisk created using the constructor marketRisk.
...     additional arguments.

Value

a numeric value, the base simulations.

See Also

simulate, scenarioRisk.

Description

split characters by presence of '\'.

Usage

splitComma(x, rm.spaces = T)

Arguments

x  a character vector.
rm.spaces a logical value, should the spaces before and after commas be deleted?

Value

a character vector.
spread

Constructing a Spread (Risk Factor)

Description

Constructor for the S3 class spread. It allows to define a spread-type risk factor. This risk factor refers to the "Modell-Spread" change $\Delta S(1,j,r)$ for a certain index rating $r$ and a certain currency $j$ in the valuation function for "Fixed-Income-Assets und Versicherungsverpflichtungen" at page 6 in the FINMA document "SST-Marktrisiko und -Aggregation Technische Beschreibung".

Usage

\[
\text{spread}(\text{name}, \text{currency}, \text{rating}, \text{scale} = \text{NULL})
\]

Arguments

- **name**: a character value of length one. This corresponds to the name in the covariance matrix of the marketRisk to which the spread risk factor is mapped. This means that the risk factor change $\Delta S(1,j,r)$ in the FINMA document "SST-Markrisiko und -Aggregation Technische Beschreibung" will be assumed to be modeled by the underlying normal random variable corresponding to name in the covariance matrix (potentially scaled by scale if not NULL).

- **currency**: a character value of length one. The currency in which the underlying "Fixed-Income-Assets oder Versicherungsverpflichtungen" is valuated. This refers to the currency corresponding to the index $j$ in the FINMA document "SST-Markrisiko und -Aggregation Technische Beschreibung" (version 31.1.2018).

- **rating**: a character value of length one. The corresponding rating of the spread refering to the index $r$ at in the FINMA document "SST-Markrisiko und -Aggregation Technische Beschreibung".

- **scale**: a numeric value of length one. If not set NULL, this defines a scaled risk factor equal to scale times the risk factor defined by name in the covariance matrix contained in marketRisk. By default its value is scale = NULL.

Value

An S3 object, instance of the class spread.

Examples

```r
# constructing a non-scaled spread risk factor
# (assuming "AA_EUR_Spread" exists in marketRisk).

e <- spread(name = "AA_EUR_Spread",
            rating = "AA",
            currency = "EUR")

# constructing a scaled spread risk factor
# (assuming "AA_EUR_Spread" exists in marketRisk).
```
e <- spread(name = "AA_EUR_Spread",
             rating = "AA",
             currency = "EUR",
             scale = 0.5)

# Spread in Object?

## Description
S3 generic to check that the spread is in the object.

## Usage
spreadIsIn(object, ...)

## Arguments
- **object**: an S3 object potentially containing the spread.
- **...**: additional parameters.

## Value
a logical value.

# Spread in Standalone?

## Description
S3 generic to check that the spread is in the object.

## Usage
## S3 method for class 'standalone'
spreadIsIn(object, currency, rating, ...)

## Arguments
- **object**: S3 object of class standalone.
- **currency**: character value. A currency.
- **rating**: character value. The rating associated to the spread.
- **...**: additional arguments.
Value

a logical value, is the spread in the standalone?

See Also

spreadIsIn.

---

### Constructing an sstModel

**Description**

Constructor for the S3 class sstModel (main class of the package). It allows to build for a Swiss Solvency Test Model (SST model aggregating risk information with a portfolio description).

**Usage**

```r
sstModel(portfolio, market.risk, life.risk = NULL, health.risk = NULL,
         nonlife.risk = NULL, scenario.risk = NULL, participation.risk = NULL,
         macro.economic.scenarios = NULL, nhmr = NULL, reordering.parameters,
         standalones = NULL)
```

**Arguments**

- **portfolio**: a portfolio S3 object.
- **market.risk**: a marketRisk S3 object.
- **life.risk**: a lifeRisk S3 object. This can be NULL in case no lifeRisk is considered.
- **health.risk**: a healthRisk S3 object. This can be NULL in case no healthRisk is considered.
- **nonlife.risk**: a nonLifeRisk S3 object. This can be NULL in case no nonLifeRisk is considered.
- **scenario.risk**: a scenarioRisk S3 object. This can be NULL in case no scenarioRisk is considered.
- **participation.risk**: a participationRisk S3 object. This can be NULL in case no participationRisk is considered.
- **macro.economic.scenarios**: a macroEconomicScenarios S3 object. This should be compatible with the portfolio and the marketRisk, please consult `?macroEconomicScenarios` for more information.
- **nhmr**: NULL or numeric value of length one and in [0, 1]. The factor for non-headgeable market risk for market value margin computation.
- **reordering.parameters**: list of reordering information containing the following fields...
• list.correlation.matrix: list of correlation matrices. The list should contain at least one correlation matrix named "base" (in first position) representing the base correlation from which ranks are simulated (with the associated Gaussian copula). If no additional correlation matrix is provided, a simple Gaussian reordering is applied. If additional named correlation matrices are provided then conditional reordering with stressed Gaussian copulas is applied. The names of the extra correlation matrices correspond to the names of the stressed-scenarios. In any case the rownames and colnames of the correlation matrices should be c("market", "life", "health", "nonlife").

• region.boundaries: matrix with colnames corresponding to the risks (respecting the prescribed order) c("market", "life", "health", "nonlife") and rownames to the scenarios names (the names of the extra correlation matrices provided in the list list.correlation.matrix). This should be NULL in the case of a simple Gaussian reordering (i.e. list.correlation.matrix contains only a single element named "base").

• region.probability a numeric value of probabilities (one for each extra scenario) giving the probability that the base Gaussian copula (represented by the correlation matrix named "base" in list.correlation.matrix) takes its values within the extreme regions (rectangles). This should be NULL in case of a simple Gaussian reordering.

• scenario.probability a numeric value of probabilities (one for each extra scenario) giving the probabilities of each scenario. This should be NULL in the case of a simple Gaussian reordering.

standalones a list of standalone S3 objects. Please note that names of standalones should not appear in base market risk factors names in market.risk.

Value
an S3 object, instance of the class sstModel.

Note
portfolio and market.risk should have the same base currency. Moreover, all risks should be consistent between them and the portfolio should be consistent with all risks. Note also that more information on the reordering can be found in the help page of the function conditionalReordering.

See Also
summary.sstModel, print.sstModel.
Usage

`sstModel_check()`

Value

A character vector containing all the packages' names.

---

`sstModel_news` *Display sstModel R-package News File*

Description

Display the NEWS.md file to obtain information about new features implemented in the packages, code optimizations, changes of API, bug fixes, etc.

Usage

`sstModel_news()`

---

`sstRatio` *Compute the Swiss Solvency Test (SST) Ratio*

Description

S3 generic method to compute the sst ratio.

Usage

`sstRatio(object, ...)`

Arguments

- `object`: an S3 object.
- `...`: additional parameters

Value

a numeric value.
**sstRatio.sstOutput**

*Compute the Swiss Solvency Test (SST) Ratio*

**Description**

S3 generic method to compute the sst ratio.

**Usage**

```r
## S3 method for class 'sstOutput'
sstRatio(object, with.scenario = F, ...)
```

**Arguments**

- `object`: S3 object of class sstOutput.
- `with.scenario`: logical value of length one. Should the target capital be computed with scenario risk also?
- `...`: additional parameters to be passed on to `marketValueMargin` and `riskCapital`.

**Value**

a numeric value. The Swiss Solvency Test Ratio.

**See Also**

`sstRatio`.

---

**standalone**

*Constructing a Standalone Market Risk*

**Description**

standalone Constructor for the S3 class standalone. A *standalone market risk* corresponds to a sub-model for market risk where only a subset of all market RiskFactors in a `marketRisk` is considered.

**Usage**

```r
standalone(name, ..., list.arg = F)
```
Arguments

name character value of length one representing the name of the standalone market risk. Please refer to the note Section to see which names cannot be used because there are reserved names for the model. Using such a name would trigger an error at the standalone construction.

... S3 objects of class riskFactor.

list.arg logical value of length one, by default set to FALSE. It allows to use ... argument to pass a list of objects of class riskFactor.

Value

a S3 object, instance of the class standalone.

Note

The following names are reserved for the model and cannot be used to name a standalone:

- marketRisk
- lifeRisk
- healthRisk
- nonLifeRisk
- scenarioRisk
- participationRisk
- participation
- marketParticipationRisk
- asset
- cashflow
- liability
- assetForward
- fxForward
- delta

See Also

summary.standalone, print.standalone.

Examples

# Creating a new standalone.
standalone1 <- standalone(name = "CHF rates",
    rate(name = "2YCHF", currency = "CHF", horizon = "k"),
    rate(name = "10YCHF", currency = "CHF", horizon = "m"),
    rate(name = "10YCHF", currency = "CHF", horizon = "1",
        scale = 0.75))
standaloneExpectedShortfall

Compute expected shortfall for standalone risk by reference

Description

S3 generic method to compute expected shortfall of a standalone risk.

Usage

standaloneExpectedShortfall(object, ...)

Arguments

object an S3 object of class sstOutput.
... additional parameters passed to expectedShortfall.

Value

a numeric value, the expected shortfall.

See Also

getDrbc

standaloneExpectedShortfall.sstOutput

Compute expected shortfall for standalone risk by reference

Description

S3 generic method to compute expected shortfall of a standalone risk.

Usage

## S3 method for class 'sstOutput'
standaloneExpectedShortfall(object, col.name, ...)

Arguments

object S3 object of class sstOutput.
col.name name of the column in object$simulations to get the expected shortfall from.
... additional arguments passed to expectedShortfall.
Value

a numeric value, the expected shortfall.

See Also

getDrbc.

summary.asset

Summarizing an Asset with Direct Market Price

Description

summary method for the S3 class asset.

Usage

## S3 method for class 'asset'
summary(object, ...)

Arguments

object S3 object of class asset.
...
additional arguments.

Value

an S3 object, instance of class c("summaryDefault", "table").

See Also

summary.asset.

Examples

# Creating an asset.
a <- asset("equity", "USD", 1000)
# summarizing the asset.
summary(a)
### Summary.assetForward

**Summary:**

Summary method for the S3 class `assetForward`.

**Usage**

```r
## S3 method for class 'assetForward'
summary(object, ...)
```

**Arguments**

- `object`: S3 object of class `assetForward`.
- `...`: Additional arguments affecting the summary produced.

**Value**

S3 object, instance of class `summaryDefault`, `table`.

**See Also**

- `summary`, `assetForward`.

**Examples**

```r
# Creating an asset forward.
af <- assetForward("equity", "EUR", 1, 1000, 1200, "long")
# Summarizing the asset forward.
summary(af)
```

### Summary.cashflow

**Summary:**

Summary method for the S3 class `cashflow`.

**Usage**

```r
## S3 method for class 'cashflow'
summary(object, ...)
```

**Examples**

```r
t # Summarizing a Fixed-Income-Asset.	summary(af)
```
Arguments

object: S3 object of class cashflow.

Value

an S3 object, instance of class \texttt{c("summaryDefault", "table")}.

See Also

\texttt{summary}, \texttt{cashflow}

Examples

# Creating a cashflow.
cashflow <- cashflow(1L, "USD", "AAA", 0.1, 1000)  # summarizing the cashflow.
summary(cashflow)

---

summary.delta \hspace{1cm} \textit{Summarizing a Delta-Normal Remainder Term}

Description

summary method for S3 class delta.

Usage

## S3 method for class 'delta'
summary(object, ...)

Arguments

object: S3 object of class delta.

Value

an S3 object, instance of class \texttt{c("summaryDefault", "table")}.

See Also

\texttt{summary.delta}.
Examples

# Creating a new delta.
delta1 <- delta(name = c("equity", "2YCHF", "EURCHF"),
    currency = c("EUR", "CHF", "EUR"),
    sensitivity = c(100, 150, 130))
# summarizing the delta.
summary(delta1)

# Creating a new fx forward.
fxf <- fxForward("USD", "EUR", 1, 1000, 1.05, "long")
# summarizing the fx forward.
summary(fxf)
summary.health

**Summarizing a Health Delta-Normal Term**

**Description**

summary method for the S3 class health.

**Usage**

```r
## S3 method for class 'health'
summary(object, ...)
```

**Arguments**

- `object`: 3 S3 object of class health.
- `...`: additional arguments affecting the summary produced.

**Value**

an S3 object, instance of class c("summaryDefault", "table").

**See Also**

`summary`, `health`.

**Examples**

```r
# Creating a new health item.
health1 <- health(name = c("pandemy", "longevity", "storno"),
currency = c("EUR", "CHF", "EUR"),
sensitivity = c(100, 150, 130))
# summarizing the health item.
summary(health1)
```

summary.healthRisk

**Summarizing a HealthRisk**

**Description**

summary method for the S3 class healthRisk.

**Usage**

```r
## S3 method for class 'healthRisk'
summary(object, ...)
```
Arguments

object  S3 object of class healthRisk.
...

Value

an S3 object, instance of class c("summaryDefault", "table").

See Also

summary, healthRisk.

Examples

# Creating a new healthRisk.

corr.mat <- diag(rep(1, 2))
colnames(corr.mat) <- c("invalidity", "longetivity")
rownames(corr.mat) <- colnames(corr.mat)

healthRisk1 <- healthRisk(corr.mat = corr.mat)
# summarizing the healthRisk.
summary(healthRisk1)

summary.liability  Summarizing an Insurance Liability

Description

summary method for the S3 class liability.

Usage

## S3 method for class 'liability'
summary(object, ...)

Arguments

object  S3 object of class liability.
...

Value

an S3 object, instance of class c("summaryDefault", "table").
Summary:

See Also

summary, liability.

Examples

# Creating a liability.
liab <- liability(1, "USD", 1000)
# summarizing the liability.
summary(liab)

# Creating a new life item.
life1 <- life(name = c("pandemy", "longetivity", "storno"),
              currency = c("EUR", "CHF", "EUR"),
              sensitivity = c(100, 150, 130))
# summarizing the life item.
summary(life1)
Summary

Description

summary method for the S3 class lifeRisk.

Usage

## S3 method for class 'lifeRisk'
summary(object, ...)

Arguments

- **object**: S3 object of class lifeRisk.
- **...**: additional arguments.

Value

an S3 object, instance of class c("summaryDefault", "table").

See Also

summary, lifeRisk.

Examples

# Creating a new lifeRisk.

corr.mat <- diag(rep(1, 2))
colnames(corr.mat) <- c("invalidity", "longevity")
rownames(corr.mat) <- colnames(corr.mat)

lifeRisk1 <- lifeRisk(corr.mat = corr.mat,
                      quantile = c(0.995, 0.995))

# summarizing the lifeRisk.
summary(lifeRisk1)
summary.marketRisk  
**Summary method for S3 class marketRisk.**

**Usage**
```r
## S3 method for class 'marketRisk'
summary(object, ...)
```

**Arguments**
- **object**: S3 object of class marketRisk.
- **...**: additional arguments affecting the summary produced.

**Value**
a table with names:
- base risk-factors: the number of base risk-factors in the marketRisk.
- scaled risk-factors: the number of scaled risk-factors in the marketRisk.
- base currency: the base currency.

**See Also**
- `summary.marketRisk`.

summary.nonLifeRisk  
**Summary method for S3 class nonLifeRisk.**

**Usage**
```r
## S3 method for class 'nonLifeRisk'
summary(object, ...)
```

**Arguments**
- **object**: S3 object of class lifeRisk.
- **...**: additional arguments affecting the summary produced.
summary.participation

Value

an S3 object, instance of class c("summaryDefault", "table").

See Also

summary, nonLifeRisk.

Examples

# Creating a new nonLifeRisk.
nonLife1 <- nonLifeRisk(type = "simulations",
                        param = list(simulations = stats::rnorm(100)),
                        currency = "CHF")

# summarizing the nonLifeRisk.
summary(nonLife1)

# Creating a new nonLifeRisk.
nonLife2 <- nonLifeRisk(type = "log-normal",
                        param = list(mu = 1, sigma = 2),
                        currency = "CHF")

# summarizing the nonLifeRisk.
summary(nonLife2)

# Creating a new nonLifeRisk.
nonLife3 <- nonLifeRisk(type = "cdf",
                        param = list(cdf = data.frame(x = c(0,1,2,3),
                                                  cdf = c(0.3,0.7,0.9, 1))),
                        currency = "CHF")

# summarizing the nonLifeRisk.
summary(nonLife3)
Value

an S3 object, instance of class c("summaryDefault", "table").

See Also

summary, participation.

Examples

# Creating a new participation.
participation1 <- participation("USD", 1000)
# summarizing the participation
summary(participation1)

summary.participationRisk

  Summarizing a participationRisk

Description

summary method for the S3 class participationRisk.

Usage

## S3 method for class 'participationRisk'
summary(object, ...)

Arguments

object             S3 object of class participationRisk.
...                additional arguments.

Value

an S3 object, instance of class c("summaryDefault", "table").

See Also

summary.participationRisk.

Examples

# Creating a new participationRisk.
pr <- participationRisk(volatility = 0.5)
# summarizing the participationRisk.
summary(pr)
Summary of a Portfolio

Description

summary method for the S3 class portfolio.

Usage

```r
# S3 method for class 'portfolio'
summary(object, ...)
```

Arguments

- `object`: S3 object of class portfolio.
- `...`: additional arguments.

Value

an S3 object, instance of class `summary.portfolio`.

See Also

`summary`, `lifeRisk`.

Examples

```r
# Creating a new portfolio.
asset1 <- asset("equity", "USD", 1000)
asset2 <- asset("hedge fund", "EUR", 2000)
life1 <- life(name = c("pandemy", "longetivity", "storno"),
currency = c("CHF", "CHF", "CHF"),
sensitivity = c(-100, -150, -130))
health1 <- health(name = c("pandemy", "longetivity", "storno"),
currency = c("CHF", "CHF", "CHF"),
sensitivity = c(100, 150, 130))
participation1 <- participation("CHF", 1000)
valid.param <- list(mvm = list(mvm.life = 2, mvm.health = 4, mvm.nonlife = 3),
rtkr = 0,
rtkg = 0,
correction.term = 2,
credit.risk = 3,
expected.insurance.result = 10^6,
expected.financial.result = 10^5)

pf <- portfolio(market.items = list(asset1, asset2),
participation.item = participation1,
life.item = life1,
health.item = health1,
base.currency = "CHF")
```
portfolio.parameters = valid.param
# summarizing the portfolio
summary(pf)

summary.scenarioRisk  Summarizing a ScenarioRisk

Description

summary method for the S3 class scenarioRisk.

Usage

## S3 method for class 'scenarioRisk'
summary(object, ...)

Arguments

- **object**: S3 object of class scenarioRisk.
- **...**: additional arguments.

Value

an S3 object, instance of class c("summaryDefault", "table").

See Also

summary, scenarioRisk.

Examples

# Creating a new scenarioRisk.
scenarios <- scenarioRisk(name = c("earthquake", "real estate crash"),
probability = c(0.001, 0.01),
currency = c("CHF", "CHF"),
effect = c(1000, 10000))
# summarizing the scenarioRisk.
summary(scenarios)
**summary.sstModel**

*Summarizing an sstModel*

**Description**

summary method for the S3 sstModel.

**Usage**

```r
## S3 method for class 'sstModel'
summary(object, ...)
```

**Arguments**

- `object` S3 object of class sstModel.
- `...` additional arguments affecting the summary produced.

**Value**

an S3 object, instance of class summary.sstModel.

**See Also**

`summary`, `sstModel`.

---

**summary.sstOutput**

*Summarizing a sstOutput*

**Description**

summary method for S3 class sstOutput.

**Usage**

```r
## S3 method for class 'sstOutput'
summary(object, ...)
```

**Arguments**

- `object` S3 object of class sstOutput.
- `...` additional arguments to be passed to `marketValueMargin`, `riskCapital`, `targetCapital`, `sstRatio`, `expectedShortfall`. It allows to modify parameters `nhmr` for market value margin computations, `alpha` and `sup` for all expected shortfall computations with `expectedShortfall`.
summary.standalone

Value

an S3 object, instance of class c("summaryDefault", "table").

See Also

summary.standalone

Summary

Summarizing a standalone

Description

summary method for the S3 class standalone.

Usage

## S3 method for class 'standalone'
summary(object, ...)

Arguments

object S3 object of class standalone.

... additional arguments affecting the summary produced.

Value

a table with names:

- name: the number of base risk-factors in the marketRisk.
- number of risk-factors: the number of risk-factors in the standalone.

See Also

summary, marketRisk.
Table 1: Table to Asset Forward

**Description**

Internal helper for parsing.

**Usage**

```r
tableToAssetForward(table)
```

**Arguments**

- `table`: a data.frame.

**Value**

A list of object of class `assetForward`.

**See Also**

- `assetForward`

Table 2: Table to Assets

**Description**

Internal helper for parsing.

**Usage**

```r
tableToAssets(table)
```

**Arguments**

- `table`: a data.frame.

**Value**

A list of object of class `asset`.

**See Also**

- `asset`
tableToCashflow  Parsing a table to a list of cashflow

Description
internal helper for parsing.

Usage
tableToCashflow(table)

Arguments
table    a data.frame.

Value
da list of object of class liability.

See Also
cashflow.

tableToFxForward  Parsing a table to a list of fxForward

Description
internal helper for parsing.

Usage
tableToFxForward(table)

Arguments
table    a data.frame.

Value
da list of object of class assetForward.

See Also
fxForward.
tableToLiability  
\textit{Parsing a table to a list of liability}

\textbf{Description}

internal helper for parsing.

\textbf{Usage}

\texttt{tableToLiability(table)}

\textbf{Arguments}

\begin{itemize}
  \item \texttt{table} \hspace{2em} a data.frame.
\end{itemize}

\textbf{Value}

a list of object of class liability.

\textbf{See Also}

\texttt{liability}.

targetCapital  
\textit{Target Capital}

\textbf{Description}

targetCapital is a generic S3 method for S3 classes from which target capital can be provided.

\textbf{Usage}

\texttt{targetCapital(object, \ldots)}

\textbf{Arguments}

\begin{itemize}
  \item \texttt{object} \hspace{2em} an S3 object from which to obtain the target capital.
  \item \texttt{\ldots} \hspace{2em} additional parameters.
\end{itemize}

\textbf{Value}

information about target capital.
targetCapital.sstOutput

*Compute the Target Capital (TC)*

**Description**

targetCapital is a generic S3 method for S3 classes from which target capital can be provided.

**Usage**

```r
# S3 method for class 'sstOutput'
targetCapital(object, with.scenario = F, ...)
```

**Arguments**

- `object`: S3 object of class sstOutput.
- `with.scenario`: logical value of length one. Should the target capital be compute with scenario risk also?
- `...`: additional parameters to be passed on to `marketValueMargin` and `riskCapital`.

**Value**

a numeric value. The target Capital (TC).

**See Also**

targetCapital.

---

translate

*translate*

**Description**

translate is a generic S3 method for translating variable names to understandable sentences.

**Usage**

```r
translate(object, ...)
```

**Arguments**

- `object`: an S3 object to translate the fields.
- `...`: additional parameters.

**Value**

a character vector.
translate.sstOutput  Translation of Fields of sstOutput

Description
translate S3 method for sstOutput. This method allow to translate code-related naming convention to human-understandable names.

Usage
## S3 method for class 'sstOutput'
translate(object, ...)

Arguments
object  S3 object of class sstOutput.
...  additional arguments.

Value
a named character vector. The values correspond to the columns of object and the names to their translation to humanly readable titles.

See Also
summary.

valExpression  Valuation Expression

Description
valExpression is a generic S3 method for S3 classes inheriting from item. It returns the valuation expression.

Usage
dvalExpression(object, ...)

Arguments
object  an S3 object from which to construct a valuation expression.
...  additional parameters.

Value
a character value.
valExpression.asset

Building the Valuation Expression for Asset with Direct Market Price

Description

valExpression is a generic S3 method for classes inheriting from item. It returns the valuation expression.

Usage

## S3 method for class 'asset'
valExpression(object, marketRisk, standalone = NULL, ...)

Arguments

- `object` S3 object of class asset.
- `marketRisk` S3 object of class marketRisk.
- `standalone` S3 object of class standalone.
- `...` additional arguments.

Value

character value of length one, the expression representing the valuation of the asset position.

See Also

valExpression, asset, marketRisk, standalone.

---

valExpression.assetForward

Building the Valuation Expression for an Index-Forward

Description

valExpression is a generic S3 method for classes inheriting from item. It returns the valuation expression.

Usage

## S3 method for class 'assetForward'
valExpression(object, marketRisk, standalone = NULL, ...

...
valExpression.cashflow

Arguments

- **object**: S3 object of class assetForward.
- **market риск**: S3 object of class marketRisk created using marketRisk.
- **standalone**: S3 object of class standalone.
- ... additional arguments.

Value

a character value. The expression representing the valuation of the index-forward position.

See Also

valExpression, assetForward.

description

valExpression is a generic S3 method for classes inheriting from item. It returns the valuation expression.

Usage

```r
## S3 method for class 'cashflow'
valExpression(object, market риск, standalone = NULL, ...)
```

Arguments

- **object**: S3 object of class cashflow.
- **market риск**: S3 object of class marketRisk created using marketRisk.
- **standalone**: S3 object of class standalone.
- ... additional arguments.

Value

a character value. The expression representing the valuation of the cashflow position.

See Also

valExpression, cashflow, marketRisk, standalone.
valExpression.delta

Building the Valuation Expression for a Market Delta-Normal Remainder Term

Description

valExpression is a generic S3 method for classes inheriting from item. It returns the valuation expression.

Usage

```r
## S3 method for class 'delta'
valExpression(object, market.risk, standalone = NULL, ...)
```

Arguments

- `object` S3 object of class delta.
- `market.risk` S3 object of class marketRisk created using marketRisk.
- `standalone` S3 object of class standalone.
- `...` additional arguments.

Value

a character value. The expression representing the valuation of the delta remainder term.

See Also

`valExpression, delta`.

---

valExpression.fxForward

Building the Valuation Expression for a FX-Forward Position

Description

valExpression is a generic S3 method for classes inheriting from item. It returns the valuation expression.

Usage

```r
## S3 method for class 'fxForward'
valExpression(object, market.risk, standalone = NULL, ...)
```
valExpression.health

Arguments

  object          S3 object of class fxForward.
  market.risk    S3 object of class marketRisk created using marketRisk.
  standalone     S3 object of class standalone.
  ...            additional arguments.

Value

  a character value. The expression representing the valuation of the fx forward position.

See Also

  valExpression, fxForward.

Description

  valExpression is a generic S3 method for classes inheriting from item. It returns the valuation expression.

Usage

  ## S3 method for class 'health'
  valExpression(object, market.risk, health.risk, ...)

Arguments

  object          S3 object of class health.
  market.risk    S3 object of class marketRisk created using the constructor marketRisk.
  health.risk    S3 object of class healthRisk created using the constructor healthRisk.
  ...            additional arguments.

Value

  a character value. The expression representing the valuation of the health item.

See Also

  valExpression, health.
valExpression.life

Building the Valuation Expression for a Life Item

Description

valExpression is a generic S3 method for classes inheriting from item. It returns the valuation expression.

Usage

```r
## S3 method for class 'life'
valExpression(object, market.risk, life.risk, ...)  
```

Arguments

- `object`: S3 object of class `liability`.
- `market.risk`: S3 object of class `marketRisk` created using the constructor `marketRisk`.
- `standalone`: S3 object of class `standalone`.
- `...`: additional arguments.

Value

a character value. The expression representing the valuation of the liability position.

See Also

- `valExpression`, `liability`, `marketRisk`, `standalone`.

valExpression liability

Building the Valuation Expression for an Insurance Liability

Description

valExpression is a generic S3 method for S3 classes inheriting from item. It returns the valuation expression.

Usage

```r
## S3 method for class 'liability'
valExpression(object, marketNrisk, standalone = NULL, ...)  
```

Arguments

- `object`: S3 object of class `liability`.
- `marketNrisk`: S3 object of class `marketRisk` created using the constructor `marketRisk`.
- `standalone`: S3 object of class `standalone`.
- `...`: additional arguments.

Value

a character value. The expression representing the valuation of the liability position.

See Also

- `valExpression`, `liability`, `marketRisk`, `standalone`. 
valFunction

Arguments

object  S3 object of class life.
market.risk  S3 object of class marketRisk created using marketRisk.
life.risk  S3 object of class lifeRisk created using lifeRisk.
...  additional arguments.

Value

a character value. The expression representing the valuation of the life item.

See Also

valExpression, life.

---

<table>
<thead>
<tr>
<th>valFunction</th>
<th>Valuation Function</th>
</tr>
</thead>
</table>

Description

valFunction is a generic S3 method for S3 classes inheriting from item. It returns the valuation function.

Usage

valFunction(object, ...)

Arguments

object  an S3 object from which to construct a valuation function.
...  additional parameters.

Value

a function.
Description

valFunction is a generic S3 method for classes inheriting from item. This method returns the valuation function of an asset with direct market price called "Aktiven mit direkt marktabhängigen Preisen" in the FINMA technical document "SST-Standardmodell Versicherungsmodell: Zielkapital" (version 31.1.2018).

Usage

```r
## S3 method for class 'asset'
valFunction(object, marketrisk, with.constant = T, ...)
```

Arguments

- `object` S3 object of class asset.
- `marketrisk` S3 object of class marketRisk created using the constructor marketRisk.
- `with.constant` a logical value, should the expression be with constant (mean zero variation) or not?
- `...` additional arguments.

Value

a function with one argument:

- `x`: a matrix of simulations (numeric values) with named columns corresponding exactly to the name of base risk-factors in marketRisk keeping the same order, or an unnamed vector of simulations (numeric values) keeping the same ordering of base risk-factors as in marketRisk.

Note

the function returns the one-year profit variation (with mean zero or not depending on with.constant).

See Also

valFunction, asset, marketRisk.
Building the Valuation Function for an Index-Forward

Description

valFunction is a generic S3 method for classes inheriting from item. It returns the valuation function.

Usage

## S3 method for class 'assetForward'
valFunction(object, marketRisk, with.constant = T, ...)  

Arguments

- **object**: S3 object of class assetForward.
- **marketRisk**: S3 object of class marketRisk created using marketRisk.
- **with.constant**: a logical value, should the expression be with constant or not?
- **...**: additional arguments.

Value

a function with one argument:

- **x**: a matrix of simulations (numeric values) with named columns corresponding exactly to the name of base risk-factors in marketRisk keeping the same order, or an unnamed vector of simulations (numeric values) keeping the same ordering of base risk-factors as in marketRisk.

See Also

valFunction, assetForward.

Building the Valuation Function for a Fixed-Income-Asset

Description

valFunction is a generic S3 method for classes inheriting from item. It returns the valuation function.

Usage

## S3 method for class 'cashflow'
valFunction(object, marketRisk, with.constant = T, ...)

- **object**: S3 object of class cashflow.
valFunction.delta

Arguments

- **object**: S3 object of class cashflow.
- **market.risk**: S3 object of class marketRisk created using marketrisk.
- **with.constant**: a logical value, should the expression be with constant or not?
- **...**: additional arguments.

Value

a function with one argument:

- **x**: a matrix of simulations (numeric values) with named columns corresponding exactly to the name of base risk-factors in marketRisk keeping the same order, or an unnamed vector of simulations (numeric values) keeping the same ordering of base risk-factors as in marketRisk.

See Also

- valFunction, cashflow, marketrisk.

valFunction.delta  Building the Valuation Function for a Market Delta-Normal Remainder Term

Description

valFunction is a generic S3 method for classes inheriting from item. It returns the valuation function.

Usage

```r
# S3 method for class 'delta'
valFunction(object, market.risk, ...)
```

Arguments

- **object**: S3 object of class delta.
- **market.risk**: S3 object of class marketRisk created using marketrisk.
- **...**: additional arguments.

Value

a function with arguments:

- **x**: a matrix of simulations (numeric values) with named columns corresponding exactly to the name of base risk-factors in marketRisk keeping the same order, or an unnamed vector of simulations (numeric values) keeping the same ordering of base risk-factors as in marketRisk.

See Also

- valFunction, delta.
valFunction.fxForward  Building the Valuation Function for a FX-Forward

Description

valFunction is a generic S3 method for classes inheriting from item. It returns the valuation function.

Usage

```r
## S3 method for class 'fxForward'
valFunction(object, market.risk, with.constant = T, ...)
```

Arguments

- `object`: S3 object of class fxForward.
- `market.risk`: S3 object of class marketRisk created using marketRisk.
- `with.constant`: a logical value, should the expression be with constant or not?
- `...`: additional arguments.

Value

a function with one argument:

- `x`: a matrix of simulations (numeric values) with named columns corresponding exactly to the name of base risk-factors in marketRisk keeping the same order, or an unnamed vector of simulations (numeric values) keeping the same ordering of base risk-factors as in marketRisk.

See Also

valFunction, fxForward.

valFunction.liability  Building the Valuation Function for an Insurance Liability Valuation

Description

valFunction is a generic S3 method for classes inheriting from item. It returns the valuation function.

Usage

```r
## S3 method for class 'liability'
valFunction(object, market.risk, with.constant = T, ...)
```
\textbf{Arguments}

- \textit{object}: S3 object of class liability.
- \textit{market-risk}: S3 object of class marketRisk created using \texttt{marketRisk}.
- \textit{with.constant}: a logical value, should the expression be with constant or not?
- ...: additional arguments.

\textbf{Value}

a function with one argument:

\begin{itemize}
  \item \texttt{x}: a matrix of simulations (numeric values) with named columns corresponding exactly to the name of base risk-factors in \texttt{marketRisk} keeping the same order, or an unnamed vector of simulations (numeric values) keeping the same ordering of base risk-factors as in \texttt{marketRisk}.
\end{itemize}

\textbf{See Also}

\texttt{valFunction}, \texttt{liability}.

---

\textbf{valInfo} \hspace{1cm} \textit{Providing Valuation Information}

\textbf{Description}

\texttt{valInfo} is a generic S3 method for S3 classes inheriting from item. It returns sufficient information for the creation of the valuation function of the item.

\textbf{Usage}

\texttt{valInfo(object, \ldots)}

\textbf{Arguments}

- \texttt{object}: an S3 object from which to extract information.
- ...: additional parameters.

\textbf{Value}

a list.
Description

valInfo is a generic S3 method for classes inheriting from item. It returns sufficient information for the creation of the valuation function of an item.

Usage

```r
## S3 method for class 'asset'
valInfo(object, market.risk, standalone = NULL, ...)
```

Arguments

- `object`: S3 object of class asset.
- `market.risk`: S3 object of class marketRisk created using the constructor `marketRisk`.
- `standalone`: S3 object of class standalone.
- `...`: additional arguments.

Value

a list with the following elements:

- `exposure`: numeric value of length one representing the exposure in the underlying asset.
- `constant`: numeric value of length one representing the constant centering the log-normal expression.
- `risk.factor`: a `data.frame` with columns:
  - `name`: character value representing the names of the base risk-factors.
  - `id`: integer value representing the positions of the base risk-factors in the covariance matrix in `marketRisk`.
  - `scale`: numeric value representing the scaling coefficients associated to the base risk-factors.

See Also

`valInfo`, `asset`, `marketRisk`, `standalone`. 
valInfo.assetForward  Providing Information for Index-Forward Valuation from a marketRisk

Description

valInfo is a generic S3 method for classes inheriting from item. It returns sufficient information for the creation of the valuation function of the item.

Usage

## S3 method for class 'assetForward'
valInfo(object, marketRisk, standalone = NULL, ...)

Arguments

- **object**: S3 object of class assetForward.
- **marketRisk**: S3 object of class marketRisk created using the constructor marketRisk.
- **standalone**: S3 object of class standalone.
- **...**: additional arguments.

Value

A list with the following elements:

- **asset.term**: an asset item. The underlying asset term in the forward contract.
- **liability.term**: a liability item. The liability term representing the forward contract cash-flow.

See Also

valInfo, assetForward, marketRisk.

---

valInfo.cashflow  Providing Information for Fixed-Income-Asset Valuation from a marketRisk

Description

valInfo is a generic S3 method for classes inheriting from item. It returns sufficient information for the creation of the valuation function of the item.

Usage

## S3 method for class 'cashflow'
valInfo(object, marketRisk, standalone = NULL, ...)

Arguments

- **object**: S3 object of class cashflow.
- **marketRisk**: S3 object of class marketRisk created using the constructor marketRisk.
- **standalone**: S3 object of class standalone.
- **...**: additional arguments.
Arguments

- **object**: S3 object of class `cashflow`.
- **market.risk**: S3 object of class `marketRisk` created using `marketRisk`.
- **standalone**: S3 object of class `standalone`.
- **...**: additional arguments.

Value

A list with the following elements:

- **exposure**: a numeric value of length one representing the nominal value of the `cashflow`.
- **constant**: a numeric value of length one representing the constant centering the log-normal expression.
- **risk.factor**: a `data.frame` with columns:
  - **name**: a character value representing the names of the base risk-factors.
  - **id**: an integer value representing the position of the base risk-factors in the covariance matrix contained in `marketRisk`.
  - **scale**: a numeric value. The scales associated to the base risk factors.

See Also

`valinfo`, `cashflow`, `marketRisk`, `standalone`.

---

**valInfo.delta**

Providing Information for Market Delta-Normal Remainder Term Valuation from a marketRisk

Description

`valInfo` is a generic S3 method for classes inheriting from `item`. It returns sufficient information for the creation of the valuation function of the item.

Usage

```r
## S3 method for class 'delta'
valInfo(object, market риск, standalone = NULL, ...)
```

Arguments

- **object**: S3 object of class `delta`.
- **market.risk**: S3 object of class `marketRisk` created using `marketRisk`.
- **standalone**: S3 object of class `standalone`.
- **...**: additional arguments.
Value

A list with the following elements:

- **sensitivity**: a numeric value. The sensitivities (in base currency) with respect to the base risk factors stored in `risk.factor`, the second element of the list.
- **risk.factor**: a `data.frame` with columns:
  - **name**: a character value. The names of the base risk factors.
  - **id**: an integer value. The position of the base risk factors in the covariance matrix in `marketRisk`.
  - **scale**: a numeric value. The scales associated to the base risk factors.

See Also

`valInfo`, `delta`, `marketRisk`.

---

Description

`valInfo` is a generic S3 method for classes inheriting from `item`. It returns sufficient information for the creation of the valuation function of the item.

Usage

```r
## S3 method for class 'fxForward'
valInfo(object, market.risk, standalone = NULL, ...)
```

Arguments

- **object**: S3 object of class `fxForward`.
- **market.risk**: S3 object of class `marketRisk` created using `marketRisk`.
- **standalone**: S3 object of class `standalone`.
- **...**: additional arguments.

Value

A list with the following elements:

- **floating.term**: a liability item. The liability term containing the fx rate risk.
- **fixed.term**: a liability item. The liability term containing the fixed exchange rate.
**valInfo.health**

Providing Information for Health Item Valuation from a marketRisk and a healthRisk

**Description**

valInfo is a generic S3 method for classes inheriting from item. It returns sufficient information for the creation of the valuation function of the item.

**Usage**

```r
## S3 method for class 'health'
valInfo(object, market.risk, health.risk, total.vola = T, ...)
```

**Arguments**

- `object`: S3 object of class health.
- `market.risk`: S3 object of class marketRisk created using the constructor `marketRisk`.
- `health.risk`: S3 object of class healthRisk created using the constructor `healthRisk`.
- `total.vola`: a logical value, by default set to TRUE. Should we return the total volatility? (otherwise the sensitivities).
- `...`: additional arguments.

**Value**

a numeric value: the aggregated volatility if `total.vola = TRUE`. Otherwise the named vector of volatilities for each health insurance risk factor.

---

**valInfo.liability**

Providing Information for Insurance Liability Valuation from a marketRisk

**Description**

valInfo is a generic S3 method for classes inheriting from item. It returns sufficient information for the creation of the valuation function of the item.

**Usage**

```r
## S3 method for class 'liability'
valInfo(object, market.risk, standalone = NULL, ...)
```
Arguments

object  S3 object of class liability.
market.risk  S3 object of class marketRisk created using the constructor marketRisk.
standalone  S3 object of class standalone.
...  additional arguments.

Value

A list with the following elements:

- exposure: numeric value of length one. The nominal value of the liability.
- constant: numeric value of length one. The constant centering the log-normal expression.
- risk.factor: a data.frame with columns:
  - name: character value. The names of the base risk factors.
  - id: integer value. The position of the base risk factors in the covariance matrix in marketRisk.
  - scale: numeric value. The scales associated to the base risk factors.

See Also

valinfo, liability, marketRisk, standalone.

---

valInfo.life

Providing Information for Life Item Valuation from a marketRisk and a lifeRisk

Description

valInfo is a generic S3 method for classes inheriting from item. It returns sufficient information for the creation of the valuation function of the item. It returns the volatilities for life risk-factor by transforming the value-at-risk sensitivities provided in the life constructor.

Usage

```r
# S3 method for class 'life'
valInfo(object, market.risk, life.risk, total.vola = T, ...)
```

Arguments

object  S3 object of class life.
market.risk  S3 object of class marketRisk created using the constructor marketRisk.
life.risk  S3 object of class lifeRisk created using lifeRisk.
total.vola  a logical value, by default set to TRUE.
...  additional arguments.
valueAtRisk

Value

a numeric value: the aggregated volatility if total.vola = TRUE. Otherwise the named vector of volatilities for each life insurance risk factor.

-valueAtRisk-  

Compute the Value-at-Risk

Description

function to compute the alpha-Value-at-Risk of a vector.

Usage

valueAtRisk(x, alpha = 0.005)

Arguments

x
  a numeric vector. The vector from which to compute the value-at-risk.

alpha
  numeric value, the alpha-Value-at-Risk, must take values between 0 and 1. Please note that we consider value-at-risk here to be equivalent to the alpha-quantiles of x.

Value

a numeric value. The value-at-risk.

volaToExpectedShortfall

Transform normal volatility in expected shortfall

Description

function to compute expected shortfall from volatility for normal random variables.

Usage

volaToExpectedShortfall(x, alpha = 0.01, sup = F, ...)

Arguments

x
  a numeric vector of positive volatilities.

alpha
  a numeric value. The alpha-Expected Shortfall, must take values between 0 and 1.

sup
  a logical value. If TRUE the function returns the upper expected shortfall and otherwise the lower. Default is set to FALSE.

...  
  additional parameters.
write.sstOutput

Value

a numeric vector, the expected shortfalls.

write.sstOutput  Writing a sstOutput into a fundamental data sheet

Description

write an sstOutput in a .xlsx file.

Usage

write.sstOutput(object, path, keep = NULL, new.names = NULL, ...)

Arguments

object  S3 object of class sstOut.
path  the complete path to the created .xlsx file.
keep  character value, by default set to NULL. The names of the columns of the field $simulations of the sstOutput to save additionally to the fundamental data sheet.
new.names  character value, replacement names for the columns to keep.
...  additional arguments to be passed on to summary.sstOutput.

Value

None (only used for side-effects).

Note

This function is an interface that writes the output of summary.sstOutput into an excel file.

See Also

summary.
aggregateRisks, 9
asset, 9, 15, 45, 47, 100, 134, 135, 170, 185, 190, 196, 201
assetForward, 10, 48, 100, 134, 136, 171, 185, 191, 197, 202
cashflow, 12, 16, 48, 101, 134, 137, 172, 186, 191, 198, 203
changeBaseCurrency, 13
check, 14, 15–19, 21–23
check.asset, 14
check.assetForward, 15
check.cashflow, 16
check.delta, 16
check.fxForward, 17
check.health, 18
check.liability, 18
check.life, 19
check.macroEconomicScenarios, 20
check.nonLifeRisk, 20
check.participation, 21
check.scenarioRisk, 22
check.standalone, 22
compute, 23, 24–29
compute.healthRisk, 24, 97
compute.lifeRisk, 24, 122
compute.macroEconomicScenarios, 25
compute.marketRisk, 26
compute.nonLifeRisk, 26, 130
compute.participationRisk, 27, 132
compute.scenarioRisk, 28, 157
compute.sstModel, 29
computeConstant, 29
conditionalReordering, 30, 165
containsHealth, 31, 32
containsHealth.sstOutput, 32
containsInsurance, 32, 33
containsInsurance.sstOutput, 33
containsLife, 33, 34
containsLife.sstOutput, 34
containsMarket, 35, 36
containsMarket.sstOutput, 36
containsNonLife, 36, 37, 39
containsNonLife.sstOutput, 36
containsParticipation, 37, 38
containsParticipation.sstOutput, 37
containsScenario, 38, 38, 39
containsScenario.sstOutput, 39
creditRisk, 39, 40
creditRisk.sstOutput, 40
currency, 40, 108, 113, 115
currencyIsIn, 41, 42
currencyIsIn.standalone, 42
delta, 17, 42, 49, 102, 134, 138, 172, 192, 198, 204
equity, 43, 108, 113, 115
equityIsIn, 45, 45
equityIsIn.standalone, 45
excelToSstModel, 46, 61, 117–119
expectedShortfall, 46
format, 47–59
format.asset, 47
format.assetForward, 47
format.cashflow, 47
format.delta, 49
format.fxForward, 49
format.health, 50
format.healthRisk, 50
format.liability, 51
format.life, 51
format.lifeRisk, 52
format.marketRisk, 53
format.nonLifeRisk, 53
format.participation, 54
format.participationRisk, 54
format.portfolio, 55
format.scenarioRisk, 55
containsMarket, 35, 35, 36
containsMarket.sstOutput, 35
containsNonLife, 36, 36, 37
containsNonLife.sstOutput, 36
containsParticipation, 37, 37, 38
containsParticipation.sstOutput, 37
containsScenario, 38, 38, 39
containsScenario.sstOutput, 39
creditRisk, 39, 40
creditRisk.sstOutput, 40
currency, 40, 108, 113, 115
currencyIsIn, 41, 42
currencyIsIn.standalone, 42
delta, 17, 42, 49, 102, 134, 138, 172, 192, 198, 204
equity, 43, 108, 113, 115
equityIsIn, 45, 45
equityIsIn.standalone, 45
excelToSstModel, 46, 61, 117–119
expectedShortfall, 46
format, 47–59
format.asset, 47
format.assetForward, 47
format.cashflow, 47
format.delta, 49
format.fxForward, 49
format.health, 50
format.healthRisk, 50
format.liability, 51
format.life, 51
format.lifeRisk, 52
format.marketRisk, 53
format.nonLifeRisk, 53
format.participation, 54
format.participationRisk, 54
format.portfolio, 55
format.scenarioRisk, 55
containsMarket, 35, 35, 36
containsMarket.sstOutput, 35
containsNonLife, 36, 36, 37
containsNonLife.sstOutput, 36
containsParticipation, 37, 37, 38
containsParticipation.sstOutput, 37
containsScenario, 38, 38, 39
containsScenario.sstOutput, 39
creditRisk, 39, 40
creditRisk.sstOutput, 40
currency, 40, 108, 113, 115
currencyIsIn, 41, 42
currencyIsIn.standalone, 42
delta, 17, 42, 49, 102, 134, 138, 172, 192, 198, 204
equity, 43, 108, 113, 115
equityIsIn, 45, 45
equityIsIn.standalone, 45
excelToSstModel, 46, 61, 117–119
expectedShortfall, 46
format, 47–59
format.asset, 47
format.assetForward, 47
format.cashflow, 47
format.delta, 49
format.fxForward, 49
format.health, 50
format.healthRisk, 50
format.liability, 51
format.life, 51
format.lifeRisk, 52
format.marketRisk, 53
format.nonLifeRisk, 53
format.participation, 54
format.participationRisk, 54
format.portfolio, 55
format.scenarioRisk, 55
containsMarket, 35, 35, 36
containsMarket.sstOutput, 35
containsNonLife, 36, 36, 37
containsNonLife.sstOutput, 36
containsParticipation, 37, 37, 38
containsParticipation.sstOutput, 37
containsScenario, 38, 38, 39
containsScenario.sstOutput, 39
creditRisk, 39, 40
creditRisk.sstOutput, 40
currency, 40, 108, 113, 115
currencyIsIn, 41, 42
currencyIsIn.standalone, 42
delta, 17, 42, 49, 102, 134, 138, 172, 192, 198, 204
equity, 43, 108, 113, 115
equityIsIn, 45, 45
equityIsIn.standalone, 45
excelToSstModel, 46, 61, 117–119
expectedShortfall, 46
format, 47–59
format.asset, 47
format.assetForward, 47
format.cashflow, 47
format.delta, 49
format.fxForward, 49
format.health, 50
format.healthRisk, 50
format.liability, 51
format.life, 51
format.lifeRisk, 52
format.marketRisk, 53
format.nonLifeRisk, 53
format.participation, 54
format.participationRisk, 54
format.portfolio, 55
format.scenarioRisk, 55
format.sstModel, 56
format.sstOutput, 57
format.standalone, 57
format.summary.portfolio, 58
format.summary.sstModel, 58
format.summary.sstOutput, 59
fxForward, 50, 59, 103, 134, 138, 173, 186, 193, 199

generateError, 61
generateExpression, 61
generateExpression.portfolio, 62
generateFunction, 63
generateFunction.portfolio, 63
getCurrencyId, 64, 65
getCurrencyId.marketRisk, 65
getCurrencyName, 65, 66
getCurrencyName.marketRisk, 66
getCurrencyScale, 67, 68
getCurrencyScale.marketRisk, 67
getDeltaid, 68, 69
getDeltaid.marketRisk, 68
getDrbc, 69, 70, 169, 170
getDrbc.sstOutput, 69
getEquityId, 70, 71
getEquityId.marketRisk, 71
getEquityName, 71, 72
getEquityName.marketRisk, 72
getEquityScale, 73, 74
getEquityScale.marketRisk, 73
getHealthId, 74
getHealthQuantile, 74
getHealthRisk, 75
getHealthRisk.sstOutput, 75
getInitialFX, 76, 77
getInitialFX.marketRisk, 76
getInitialRate, 77, 78
getInitialRate.marketRisk, 77
getInitialSpread, 78
getInsuranceRisk, 75, 78, 79, 82, 86
getInsuranceRisk.sstOutput, 79
getLifeId, 79, 80
getLifeId.lifeRisk, 80
getLifeQuantile, 80, 81
getLifeQuantile.lifeRisk, 81
getLifeRisk, 81
getLifeRisk.sstOutput, 82
getMappingTime, 82, 83
getMappingTime.marketRisk, 83
getMarketParticipationRisk, 83
getMarketParticipationRisk.sstOutput, 84
getMarketRisk, 84, 85, 85
getMarketRisk.sstOutput, 85
getNonLifeRisk, 86
getNonLifeRisk.sstOutput, 86
getParticipation, 87
getParticipation.sstOutput, 87
getRateId, 88, 89
getRateId.marketRisk, 88
getRateName, 89, 90
getRateName.marketRisk, 89
getRateScale, 90, 91
getRateScale.marketRisk, 90
getScenarioRisk, 87, 91, 92
getScenarioRisk.sstOutput, 92
getSpreadId, 92, 93
getSpreadId.marketRisk, 93
getSpreadName, 93, 94
getSpreadName.marketRisk, 94
getSpreadScale, 95, 96
getSpreadScale.marketRisk, 95

health, 18, 50, 96, 103, 134, 139, 174, 193
healthRisk, 24, 51, 97, 104, 140, 157, 175

initialFX, 98, 126
initialRate, 98, 126
initialSpread, 99
intToGroups, 99
is.asset, 100
is.assetForward, 100
is.cashflow, 101
is.currency, 101
is.delta, 102
is.equity, 102
is.fxForward, 103
is.health, 103
is.healthRisk, 104
is.insuranceItem, 104
is.insuranceRisk, 105
is.item, 105
is.liability, 106
is.life, 106
is.lifeRisk, 107
is.macroEconomicScenarios, 107
is.mappingTable, 108
is.marketItem, 108
### INDEX

- **is.marketRisk**, 109  
- **is.nonLifeRisk**, 109  
- **is.participation**, 110  
- **is.participationRisk**, 110  
- **is.pcRate**, 111  
- **is.portfolio**, 111  
- **is.rate**, 112  
- **is.risk**, 112  
- **is.riskFactor**, 101, 102, 111, 112, 113, 114  
- **is.scenarioRisk**, 113  
- **is.spread**, 114  
- **is.sstModel**, 114  
- **is.sstOutput**, 115  
- **is.standalone**, 115  
- **itemListToExpression**, 116  
- **itemListToFunction**, 116

- **keywordToTable**, 117  
- **keywordToTransposedTable**, 118  
- **keywordToValue**, 118

- **launchDashboard**, 119  
- **liability**, 19, 51, 106, 120, 134, 141, 176, 187, 194, 200, 206  
- **life**, 19, 52, 106, 121, 134, 142, 176, 195  
- **lifeRisk**, 25, 52, 80, 81, 107, 122, 142, 158, 177, 181

- **logNormalExpression**, 123

- **macroEconomicScenarios**, 123  
- **mappingTable**, 124, 126  
- **mappingTime**, 125, 126  
- **marketRisk**, 15–17, 19, 26, 53, 98, 99, 109, 125, 125, 143, 159, 178, 184, 190, 191, 194, 196, 198, 201–204, 206  
- **marketValueMargin**, 8, 126, 127  
- **marketValueMargin.sstOutput**, 127  
- **mvmLife**, 127

- **na.rm**, 128  
- **newtonRaphson**, 128  
- **nonLifeRisk**, 21, 27, 54, 109, 129, 134, 144, 159, 179

- **participation**, 21, 28, 54, 110, 131, 134, 145, 146, 160, 180  
- **participationRisk**, 28, 55, 132, 146, 160, 180  
- **pcRate**, 108, 113, 115, 132  
- **portfolio**, 55, 63, 64, 111, 133  
- **print**, 135–146, 148–151  
- **print.asset**, 10, 135  
- **print.assetForward**, 12, 136  
- **print.cashflow**, 13, 137  
- **print.delta**, 43, 137  
- **print.fxForward**, 60, 138  
- **print.health**, 96, 139  
- **print.healthRisk**, 97, 140  
- **print.liability**, 120, 141  
- **print.life**, 121, 141  
- **print.lifeRisk**, 122, 142  
- **print.marketRisk**, 143  
- **print.nonLifeRisk**, 130, 144  
- **print.participation**, 131, 145  
- **print.participationRisk**, 132, 145  
- **print.portfolio**, 134, 146  
- **print.scenarioRisk**, 147, 157  
- **print.sstModel**, 148, 165  
- **print.sstOutput**, 149  
- **print.standalone**, 149, 168  
- **print.summary.portfolio**, 150  
- **print.summary.sstModel**, 150  
- **print.summary.sstOutput**, 151

- **rate**, 108, 113, 115, 151  
- **rateIsIn**, 153, 154  
- **rateIsIn.standalone**, 153  
- **removePerfectCorr**, 154  
- **riskCapital**, 8, 155, 155  
- **riskCapital.standalone**, 155  
- **riskFactorToExpression**, 156

- **scenarioRisk**, 22, 28, 56, 113, 134, 148, 156, 161, 182  
- **simulate**, 157–161  
- **simulate.healthRisk**, 157  
- **simulate.lifeRisk**, 122, 158  
- **simulate.marketRisk**, 158  
- **simulate.nonLifeRisk**, 130, 159  
- **simulate.participationRisk**, 160  
- **simulate.scenarioRisk**, 157, 160  
- **splitComma**, 161  
- **spread**, 108, 113, 115, 162  
- **spreadIsIn**, 163, 164  
- **spreadIsIn.standalone**, 163  
- **sstModel**, 8, 29, 46, 56, 114, 148, 164, 183  
- **sstModel-package**, 8  
- **sstModel_check**, 165
sstModel_news, 166
sstRatio, 8, 166, 167
sstRatio.sstOutput, 167
standalone, 23, 58, 150, 167, 190, 191, 194, 201, 203, 206
standaloneExpectedShortfall, 169
standaloneExpectedShortfall.sstOutput, 169
summary, 170–184, 189, 208
summary.asset, 10, 170
summary.assetForward, 12, 171
summary.cashflow, 13, 171
summary.delta, 43, 172
summary.fxForward, 60, 173
summary.health, 96, 174
summary.healthRisk, 97, 174
summary.liability, 120, 175
summary.life, 121, 176
summary.lifeRisk, 122, 177
summary.marketRisk, 178
summary.nonLifeRisk, 130, 178
summary.participation, 131, 179
summary.participationRisk, 132, 180
summary.portfolio, 134, 181
summary.scenarioRisk, 157, 182
summary.sstModel, 165, 183
summary.sstOutput, 183
summary.standalone, 168, 184
tableToAssetForward, 185
tableToAssets, 185
tableToCashflow, 186
tableToFxForward, 186
tableToLiability, 187
targetCapital, 8, 187, 188
targetCapital.sstOutput, 188
translate, 188
translate.sstOutput, 189
valExpression, 189, 190–195
valExpression.asset, 190
valExpression.assetForward, 190
valExpression.cashflow, 191
valExpression.delta, 192
valExpression.fxForward, 192
valExpression.health, 193
valExpression.liability, 194
valExpression.life, 194
valFunction, 195, 196–200
valFunction.asset, 196
valFunction.assetForward, 197
valFunction.cashflow, 197
valFunction.delta, 198
valFunction.fxForward, 199
valFunction.liability, 199
valInfo, 200, 201–204, 206
valInfo.asset, 201
valInfo.assetForward, 202
valInfo.cashflow, 202
valInfo.delta, 203
valInfo.fxForward, 204
valInfo.health, 205
valInfo.liability, 205
valInfo.life, 206
valueAtRisk, 207
volaToExpectedShortfall, 207
write.sstOutput, 208