# Package 'RQuantLib'

January 26, 2010

**Title** R interface to the QuantLib library

**Version** 0.3.2

**Date** $Date: 2010-01-14 23:09:45 -0500 (Thu, 14 Jan 2010) $

**Maintainer** Dirk Eddelbuettel <edd@debian.org>

**Author** Dirk Eddelbuettel <edd@debian.org> and Khanh Nguyen <knguyen@cs.umb.edu>

**Description** The RQuantLib package makes parts of QuantLib visible to the R user. Currently a number option pricing functions are included, both vanilla and exotic, as well as a broad range of fixed-income functions, as well a general calendaring and holiday utilities. Further software contributions are welcome.

The QuantLib project aims to provide a comprehensive software framework for quantitative finance. The goal is to provide a standard open source library for quantitative analysis, modeling, trading, and risk management of financial assets.

The Windows binary version is self-contained and does not require a QuantLib (or Boost) installation.

RQuantLib uses the Rcpp R/C++ interface class library. See the Rcpp package on CRAN (or R-Forge) for more information on Rcpp.

Note that while RQuantLib's code is licensed under the GPL (v2 or later), QuantLib itself is released under a somewhat less restrictive Open Source license (see QuantLib-License.txt).

**Depends** R (>= 2.7.0), Rcpp (>= 0.7.0)

**SystemRequirements** QuantLib library (>= 0.9.9) from http://quantlib.org, Boost library (>= 1.34.0) from http://www.boost.org

**License** GPL (>= 2)

**URL** http://quantlib.org
http://dirk.eddelbuettel.com/code/rquantlib.html

# R **topics documented:**

---

adjust *Calendar functions from QuantLib*

---

### Description

The `adjust` function evaluates the given dates in the context of the given calendar, and returns a vector that adjusts each input dates to the appropriate near business day with respect to the given convention. .

### Usage

```
adjust(calendar="TARGET", dates=Sys.Date(), bdc = 0)
```

### Arguments

calendar   A string identifying one of the supported QuantLib calendars, see Details for
           more

dates      A vector (or scalar) of `Date` types.

bdc        business day convention. By default, this value is 0 and correspond to Following
           convention

### Details

The calendars are coming from QuantLib, and the QuantLib documentation should be consulted for details.

Currently, the following strings are recognised: TARGET (a default calendar), Canada and Canada/Settlement, Canada/TSX, Germany and Germany/FrankfurtStockExchange, Germany/Settlement, Germany/Xetra, Germany/Eurex, Italy and Italy/Settlement, Italy/Exchange, Japan, UnitedKingdom and UnitedKingdom/Settlement, UnitedKingdom/Exchange, UnitedKingdom/Metals, UnitedStates and UnitedStates/Settlement, UnitedStates/NYSE, UnitedStates/GovernmentBond, UnitedStates/NERC.

(In case of multiples entries per country, the country default is listed right after the country itself. Using the shorter form is equivalent.)

### Value

An named vector of dates. The element names are the dates (formatted as text in yyyy-mm-dd format).

### Note

The interface might change in future release as `QuantLib` stabilises its own API.

### Author(s)

Dirk Eddelbuettel <edd@debian.org> for the R interface; Khanh Nguyen <nguyen.h.khanh@gmail.com> for the implementation; the QuantLib Group for `QuantLib`

## References

http://quantlib.org for details on QuantLib.

## Examples

```
dates <- seq(from=as.Date("2009-04-07"), to=as.Date("2009-04-14"), by=1)
adjust("UnitedStates", dates)
adjust("UnitedStates/Settlement", dates)      ## same as previous
adjust("UnitedStates/NYSE", dates)            ## stocks
adjust("UnitedStates/GovernmentBond", dates)  ## bonds
adjust("UnitedStates/NERC", dates)            ## energy
```

---

advance                          *Calendar functions from QuantLib*

---

## Description

The advance function evaluates the given dates in the context of the given calendar, and returns a vector that advances the given dates of the given number of business days and returns the result. **See note for usage below for usage.**

## Arguments

| | |
|---|---|
| calendar | A string identifying one of the supported QuantLib calendars, see Details for more |
| dates | A vector (or scalar) of Date types. |
| n | an int |
| timeUnit | a value of 0,1,2,3 that corresponds to Days, Weeks, Months, and Year. For more detail, see QuantLib doc http://quantlib.org/reference/group_ _datetime.html |
| period | See Enum |
| bdc | business day convention. By default, this value is 0 and correspond to Following convention |
| emr | End Of Month rule. Default = false |

## Details

The calendars are coming from QuantLib, and the QuantLib documentation should be consulted for details.

Currently, the following strings are recognised: TARGET (a default calendar), Canada and Canada/Settlement, Canada/TSX, Germany and Germany/FrankfurtStockExchange, Germany/Settlement, Germany/Xetra, Germany/Eurex, Italy and Italy/Settlement, Italy/Exchange, Japan, UnitedKingdom and United-Kingdom/Settlement, UnitedKingdom/Exchange, UnitedKingdom/Metals, UnitedStates and UnitedStates/Settlement, UnitedStates/NYSE, UnitedStates/GovernmentBond, UnitedStates/NERC.

(In case of multiples entries per country, the country default is listed right after the country itself. Using the shorter form is equivalent.)

## Value

An named vector of dates. The element names are the dates (formatted as text in yyyy-mm-dd format).

## Note for usage

The function can be called in two ways.

advance(calendar="TARGET", dates=Sys.Date(),n, timeUnit, bdc = 0, emr =0)

advance(calendar="TARGET", dates=Sys.Date(), period, bdc = 0, emr =0)

## Author(s)

Dirk Eddelbuettel <edd@debian.org> for the R interface; Khanh Nguyen <nguyen.h.khanh@gmail.com> for the implementation; the QuantLib Group for QuantLib

## References

http://quantlib.org for details on QuantLib.

## Examples

```
dates <- seq(from=as.Date("2009-04-07"), to=as.Date("2009-04-14"), by=1)
advance("UnitedStates", dates, 10, 0)
advance("UnitedStates/Settlement", dates, 10, 1)      ## same as previous
advance("UnitedStates/NYSE", dates, 10, 2)            ## stocks
advance("UnitedStates/GovernmentBond", dates,  10, 3)  ## bonds
advance("UnitedStates/NERC", dates, period = 3)            ## energy
```

---

AmericanOption          *American Option evaluation using Finite Differences*

---

## Description

This function evaluations an American-style option on a common stock using finite differences. The option value as well as the common first derivatives ("Greeks") are returned.

## Usage

```
## Default S3 method:
AmericanOption(type, underlying, strike, dividendYield, riskFreeRate,
maturity, volatility, timeSteps=150, gridPoints=151)

## S3 method for class 'Option':
print
## S3 method for class 'Option':
summary
```

**Arguments**

| | |
|---|---|
| type | A string with one of the values `call` or `put` |
| underlying | Current price of the underlying stock |
| strike | Strike price of the option |
| dividendYield | |
| | Continuous dividend yield (as a fraction) of the stock |
| riskFreeRate | Risk-free rate |
| maturity | Time to maturity (in fractional years) |
| volatility | Volatility of the underlying stock |
| timeSteps | Time steps for the Finite Differences method, default value is 150 |
| gridPoints | Grid points for the Finite Differences method, default value is 151 |

**Details**

The Finite Differences method is used to value the American Option.

Please see any decent Finance textbook for background reading, and the `QuantLib` documentation for details on the `QuantLib` implementation.

**Value**

An object of class `AmericanOption` (which inherits from class [`Option`]) is returned. It contains a list with the following components:

| | |
|---|---|
| value | Value of option |
| delta | Sensitivity of the option value for a change in the underlying |
| gamma | Sensitivity of the option delta for a change in the underlying |
| vega | Sensitivity of the option value for a change in the underlying's volatility |
| theta | Sensitivity of the option value for a change in t, the remaining time to maturity |
| rho | Sensitivity of the option value for a change in the risk-free interest rate |
| dividendRho | Sensitivity of the option value for a change in the dividend yield |
| parameters | List with parameters with which object was created |

Note that under the new pricing framework used in QuantLib, binary pricers do not provide analytics for 'Greeks'. This is expected to be addressed in future releases of QuantLib.

**Note**

The interface might change in future release as `QuantLib` stabilises its own API.

**Author(s)**

Dirk Eddelbuettel <edd@debian.org> for the R interface; the QuantLib Group for `QuantLib`

**References**

http://quantlib.org for details on `QuantLib`.

## See Also

EuropeanOption

## Examples

```
# simple call with unnamed parameters
AmericanOption("call", 100, 100, 0.02, 0.03, 0.5, 0.4)
# simple call with some explicit parameters
AmericanOption("put", strike=100, volatility=0.4, 100, 0.02, 0.03, 0.5)
```

---

AmericanOptionImpliedVolatility

*Implied Volatility calculation for American Option*

---

## Description

The `AmericanOptionImpliedVolatility` function solves for the (unobservable) implied volatility, given an option price as well as the other required parameters to value an option.

## Usage

```
## Default S3 method:
AmericanOptionImpliedVolatility(type, value,
underlying, strike,dividendYield, riskFreeRate, maturity, volatility,
        timeSteps=150, gridPoints=151)

## S3 method for class 'ImpliedVolatility':
print
## S3 method for class 'ImpliedVolatility':
summary
```

## Arguments

| | |
|---|---|
| type | A string with one of the values call or put |
| value | Value of the option (used only for ImpliedVolatility calculation) |
| underlying | Current price of the underlying stock |
| strike | Strike price of the option |
| dividendYield | |
| | Continuous dividend yield (as a fraction) of the stock |
| riskFreeRate | Risk-free rate |
| maturity | Time to maturity (in fractional years) |
| volatility | Initial guess for the volatility of the underlying stock |
| timeSteps | Time steps for the Finite Differences method, default value is 150 |
| gridPoints | Grid points for the Finite Differences method, default value is 151 |

## Details

The Finite Differences method is used to value the American Option. Implied volatilities are then calculated numerically.

Please see any decent Finance textbook for background reading, and the QuantLib documentation for details on the QuantLib implementation.

## Value

The AmericanOptionImpliedVolatility function returns an object of class ImpliedVolatility. It contains a list with the following elements:

impliedVol    The volatility implied by the given market prices

parameters    List with the option parameters used

## Note

The interface might change in future release as QuantLib stabilises its own API.

## Author(s)

Dirk Eddelbuettel <edd@debian.org> for the R interface; the QuantLib Group for QuantLib

## References

http://quantlib.org for details on QuantLib.

## See Also

EuropeanOption,AmericanOption,BinaryOption

## Examples

```
AmericanOptionImpliedVolatility(type="call", value=11.10, underlying=100,
strike=100, dividendYield=0.01, riskFreeRate=0.03,
maturity=0.5, volatility=0.4)
```

---

AsianOption    *Asian Option evaluation using Closed-Form solution*

---

## Description

The AsianOption function evaluates an Asian-style option on a common stock using an analytic solution for continuous geometric average price. The option value, the common first derivatives ("Greeks") as well as the calling parameters are returned.

## Usage

```
## Default S3 method:
AsianOption(averageType, type, underlying, strike,
                dividendYield, riskFreeRate, maturity,
                volatility, timeSteps=150, gridPoints=151)
## S3 method for class 'Option':
plot
## S3 method for class 'Option':
print
## S3 method for class 'Option':
summary
```

## Arguments

| | |
|---|---|
| `averageType` | Specifiy averaging type, either "geometric" or "arithmetic" |
| `type` | A string with one of the values `call` or `put` |
| `underlying` | Current price of the underlying stock |
| `strike` | Strike price of the option |
| `dividendYield` | |
| | Continuous dividend yield (as a fraction) of the stock |
| `riskFreeRate` | Risk-free rate |
| `maturity` | Time to maturity (in fractional years) |
| `volatility` | Volatility of the underlying stock |
| `timeSteps` | Time steps for the Finite Differences method, default value is 150 |
| `gridPoints` | Grid points for the Finite Differences method, default value is 151 |

## Details

The well-known closed-form solution derived by Black, Scholes and Merton is used for valuation. Implied volatilities are calculated numerically.

Please see any decent Finance textbook for background reading, and the `QuantLib` documentation for details on the `QuantLib` implementation.

## Value

The `AsianOption` function returns an object of class `AsianOption` (which inherits from class `Option`). It contains a list with the following components:

| | |
|---|---|
| `value` | Value of option |
| `delta` | Sensitivity of the option value for a change in the underlying |
| `gamma` | Sensitivity of the option delta for a change in the underlying |
| `vega` | Sensitivity of the option value for a change in the underlying's volatility |
| `theta` | Sensitivity of the option value for a change in t, the remaining time to maturity |
| `rho` | Sensitivity of the option value for a change in the risk-free interest rate |
| `dividendRho` | Sensitivity of the option value for a change in the dividend yield |
| `parameters` | List with parameters with which object was created |

## Note

The interface might change in future release as `QuantLib` stabilises its own API.

## Author(s)

Dirk Eddelbuettel `<edd@debian.org>` for the R interface; the QuantLib Group for `QuantLib`

## References

`http://quantlib.org` for details on `QuantLib`.

## Examples

```
# simple call with some explicit parameters, and slightly increased vol:
AsianOption("geometric", "put", underlying=80, strike=85, div=-0.03, riskFree=0.05, maturity
```

---

BarrierOption            *Barrier Option evaluation using Closed-Form solution*

---

## Description

This function evaluations an Barrier option on a common stock using a closed-form solution. The option value as well as the common first derivatives ("Greeks") are returned.

## Usage

```
## Default S3 method:
BarrierOption(barrType, type, underlying, strike,
                          dividendYield, riskFreeRate, maturity,
                          volatility, barrier, rebate=0.0)
## S3 method for class 'Option':
print
## S3 method for class 'Option':
summary
```

## Arguments

| | |
|---|---|
| barrType | A string with one of the values `downin`, `downout`, `upin` or `upout` |
| type | A string with one of the values `call` or `put` |
| underlying | Current price of the underlying stock |
| strike | Strike price of the option |
| dividendYield | |
| | Continuous dividend yield (as a fraction) of the stock |
| riskFreeRate | Risk-free rate |
| maturity | Time to maturity (in fractional years) |

| volatility | Volatility of the underlying stock |
| barrier | Option barrier value |
| rebate | Optional option rebate, defaults to 0.0 |

### Details

A closed-form solution is used to value the Barrier Option. In the case of Barrier options, the calculations are from Haug's "Option pricing formulas" book (McGraw-Hill).

Please see any decent Finance textbook for background reading, and the QuantLib documentation for details on the QuantLib implementation.

### Value

An object of class BarrierOption (which inherits from class Option) is returned. It contains a list with the following components:

| value | Value of option |
| delta | Sensitivity of the option value for a change in the underlying |
| gamma | Sensitivity of the option delta for a change in the underlying |
| vega | Sensitivity of the option value for a change in the underlying's volatility |
| theta | Sensitivity of the option value for a change in t, the remaining time to maturity |
| rho | Sensitivity of the option value for a change in the risk-free interest rate |
| dividendRho | Sensitivity of the option value for a change in the dividend yield |
| parameters | List with parameters with which object was created |

.

Note that under the new pricing framework used in QuantLib, binary pricers do not provide analytics for 'Greeks'. This is expected to be addressed in future releases of QuantLib.

### Note

The interface might change in future release as QuantLib stabilises its own API.

### Author(s)

Dirk Eddelbuettel <edd@debian.org> for the R interface; the QuantLib Group for QuantLib

### References

http://quantlib.org for details on QuantLib.

### See Also

AmericanOption,EuropeanOption

## Examples

```
BarrierOption(barrType="downin", type="call", underlying=100,
strike=100, dividendYield=0.02, riskFreeRate=0.03,
maturity=0.5, volatility=0.4, barrier=90)
```

---

BermudanSwaption          *Bermudan swaption valuation using several short-rate models*

---

## Description

BermudanSwaption prices a Bermudan swaption with specified strike and maturity (in years), after calibrating the selected short-rate model to an input swaption volatility matrix. Swaption maturities are in years down the rows, and swap tenors are in years along the columns, in the usual fashion. It is assumed that the Bermudan swaption is exercisable on each reset date of the underlying swaps.

## Usage

```
BermudanSwaption(params, tsQuotes, swaptionMaturities, swapTenors,
volMatrix)
```

## Arguments

params          A list specifying the tradeDate (month/day/year), settlementDate, payFixed
                flag, strike, pricing method, and curve construction options (see *Examples*
                section below). Curve construction options are interpWhat (possible values
                are discount, forward, and zero) and interpHow (possible values are
                linear, loglinear, and spline). Both interpWhat and interpHow
                are ignored when a flat yield curve is requested, but they must be present never-
                theless. The pricing method can be one of the following (all short-rate models):

|            |                                              |
|------------|----------------------------------------------|
| G2Analytic | G2 2-factor Gaussian model using analytic formulas. |
| HWAnalytic | Hull-White model using analytic formulas.    |
| HWTree     | Hull-White model using a tree.               |
| BKTree     | Black-Karasinski model using a tree.         |

tsQuotes        Market observables needed to construct the spot term structure of interest rates.
                A list of name/value pairs. See the help page for DiscountCurve for details.

swaptionMaturities
                A vector containing the swaption maturities associated with the rows of the
                swaption volatility matrix.

swapTenors      A vector containing the underlying swap tenors associated with the columns of
                the swaption volatility matrix.

volMatrix       The swaption volatility matrix. Must be a 2D matrix stored by rows. See the
                example below.

**Details**

This function is based on `QuantLib` Version 0.3.10. It introduces support for fixed-income instruments in `RQuantLib`.

At present only a small number of the many parameters that can be set in `QuantLib` are exposed by this function. Some of the hard-coded parameters that apply to the current version include: day-count conventions, fixing days (2), index (Euribor), fixed leg frequency (annual), and floating leg frequency (semi-annual). Also, it is assumed that the swaption volatility matrix corresponds to expiration dates and tenors that are measured in years (a 6-month expiration date is not currently supported, for example).

Given the number of parameters that must be specified and the care with which they must be specified (with no defaults), it is not practical to use this function in the usual interactive fashion.

The simplest approach is simply to save the example below to a file, edit as desired, and `source` the result. Alternatively, the input commands can be kept in a script file (under Windows) or an Emacs/ESS session (under Linux), and selected parts of the script can be executed in the usual way.

Fortunately, the C++ exception mechanism seems to work well with the R interface, and `QuantLib` exceptions are propagated back to the R user, usually with a message that indicates what went wrong. (The first part of the message contains technical information about the precise location of the problem in the `QuantLib` code. Scroll to the end to find information that is meaningful to the R user.)

**Value**

`BermudanSwaption` returns a list containing calibrated model paramters (what parameters are returned depends on the model selected) along with:

| | |
|---|---|
| `price` | Price of swaption in basis points (actual price equals `price` times notional divided by 10,000) |
| `ATMStrike` | At-the-money strike |
| `params` | Input parameter list |

**Author(s)**

Dominick Samperi

**References**

Brigo, D. and Mercurio, F. (2001) *Interest Rate Models: Theory and Practice*, Springer-Verlag, New York.

For information about `QuantLib` see http://quantlib.org.

For information about `RQuantLib` see http://dirk.eddelbuettel.com/code/rquantlib.html.

**See Also**

DiscountCurve

## Examples

```
# This data is taken from sample code shipped with QuantLib 0.3.10.
params <- list(tradeDate=as.Date('2002-2-15'),
               settleDate=as.Date('2002-2-19'),
               payFixed=TRUE,
      strike=.06,
      method="G2Analytic",
               interpWhat="discount",
               interpHow="loglinear")

# Market data used to construct the term structure of interest rates
tsQuotes <- list(d1w  =0.0382,
                 d1m  =0.0372,
                 fut1=96.2875,
                 fut2=96.7875,
                 fut3=96.9875,
                 fut4=96.6875,
                 fut5=96.4875,
                 fut6=96.3875,
                 fut7=96.2875,
                 fut8=96.0875,
                 s3y  =0.0398,
                 s5y  =0.0443,
                 s10y =0.05165,
                 s15y =0.055175)

# Use this to compare with the Bermudan swaption example from QuantLib
#tsQuotes <- list(flat=0.04875825)

# Swaption volatility matrix with corresponding maturities and tenors
swaptionMaturities <- c(1,2,3,4,5)

swapTenors <- c(1,2,3,4,5)

volMatrix <- matrix(
                    c(0.1490, 0.1340, 0.1228, 0.1189, 0.1148,
                      0.1290, 0.1201, 0.1146, 0.1108, 0.1040,
                      0.1149, 0.1112, 0.1070, 0.1010, 0.0957,
                      0.1047, 0.1021, 0.0980, 0.0951, 0.1270,
                      0.1000, 0.0950, 0.0900, 0.1230, 0.1160),
                    ncol=5, byrow=TRUE)

# Price the Bermudan swaption
pricing <- BermudanSwaption(params, tsQuotes,
                            swaptionMaturities, swapTenors, volMatrix)
summary(pricing)
```

---

BinaryOption          *Binary Option evaluation using Closed-Form solution*

---

## Description

This function evaluations an Binary option on a common stock using a closed-form solution. The option value as well as the common first derivatives ("Greeks") are returned.

## Usage

```
## Default S3 method:
BinaryOption(binType, type, excType, underlying,
        strike, dividendYield,
riskFreeRate, maturity, volatility, cashPayoff)

## S3 method for class 'Option':
print
## S3 method for class 'Option':
summary
```

## Arguments

| | |
|---|---|
| `binType` | A string with one of the values `cash`, `asset` or `gap` to select CashOrNothing, AssetOrNothing or Gap payoff profiles |
| `type` | A string with one of the values `call` or `put` |
| `excType` | A string with one of the values `european` or `american` to denote the exercise type |
| `underlying` | Current price of the underlying stock |
| `strike` | Strike price of the option |
| `dividendYield` | |
| | Continuous dividend yield (as a fraction) of the stock |
| `riskFreeRate` | Risk-free rate |
| `maturity` | Time to maturity (in fractional years) |
| `volatility` | Volatility of the underlying stock |
| `cashPayoff` | Payout amount |

## Details

A closed-form solution is used to value the Binary Option.

Please see any decent Finance textbook for background reading, and the `QuantLib` documentation for details on the `QuantLib` implementation.

## Value

An object of class `BinaryOption` (which inherits from class [Option](#)) is returned. It contains a list with the following components:

| | |
|---|---|
| `value` | Value of option |
| `delta` | Sensitivity of the option value for a change in the underlying |
| `gamma` | Sensitivity of the option delta for a change in the underlying |

| vega | Sensitivity of the option value for a change in the underlying's volatility |
| theta | Sensitivity of the option value for a change in t, the remaining time to maturity |
| rho | Sensitivity of the option value for a change in the risk-free interest rate |
| dividendRho | Sensitivity of the option value for a change in the dividend yield |
| parameters | List with parameters with which object was created |

### Note

The interface might change in future release as QuantLib stabilises its own API.

### Author(s)

Dirk Eddelbuettel <edd@debian.org> for the R interface; the QuantLib Group for QuantLib

### References

http://quantlib.org for details on QuantLib.

### See Also

AmericanOption,EuropeanOption

### Examples

```
BinaryOption(binType="asset", type="call", excType="european", underlying=100, strike=100, d
               riskFreeRate=0.03, maturity=0.5, volatility=0.4, cashPayoff=10)
```

---

BinaryOptionImpliedVolatility

*Implied Volatility calculation for Binary Option*

---

### Description

The BinaryOptionImpliedVolatility function solves for the (unobservable) implied volatility, given an option price as well as the other required parameters to value an option.

### Usage

```
## Default S3 method:
BinaryOptionImpliedVolatility(type, value, underlying,
strike, dividendYield, riskFreeRate, maturity, volatility,
cashPayoff=1)

## S3 method for class 'ImpliedVolatility':
print
## S3 method for class 'ImpliedVolatility':
summary
```

## Arguments

| | |
|---|---|
| `type` | A string with one of the values `call`, `put` or `straddle` |
| `value` | Value of the option (used only for ImpliedVolatility calculation) |
| `underlying` | Current price of the underlying stock |
| `strike` | Strike price of the option |
| `dividendYield` | |
| | Continuous dividend yield (as a fraction) of the stock |
| `riskFreeRate` | Risk-free rate |
| `maturity` | Time to maturity (in fractional years) |
| `volatility` | Initial guess for the volatility of the underlying stock |
| `cashPayoff` | Binary payout if options is exercised, default is 1 |

## Details

The Finite Differences method is used to value the Binary Option. Implied volatilities are then calculated numerically.

Please see any decent Finance textbook for background reading, and the `QuantLib` documentation for details on the `QuantLib` implementation.

## Value

The `BinaryOptionImpliedVolatility` function returns an object of class `ImpliedVolatility`. It contains a list with the following elements:

| | |
|---|---|
| `impliedVol` | The volatility implied by the given market prices |
| `parameters` | List with the option parameters used |

## Note

The interface might change in future release as `QuantLib` stabilises its own API.

## Author(s)

Dirk Eddelbuettel <edd@debian.org> for the R interface; the QuantLib Group for `QuantLib`

## References

http://quantlib.org for details on `QuantLib`.

## See Also

`EuropeanOption`,`AmericanOption`,`BinaryOption`

## Examples

```
BinaryOptionImpliedVolatility("call", value=4.50, strike=100, 100, 0.02, 0.03, 0.5, 0.4, 10)
```

---

Bond                                         *Base class for Bond price evalution*

---

### Description

This class forms the basis from which the more specific classes are derived.

### Usage

```
## S3 method for class 'Bond':
print
## S3 method for class 'Bond':
plot
## S3 method for class 'Bond':
summary
```

### Arguments

Bond              Any Bond object derived from this base class

### Details

Please see any decent Finance textbook for background reading, and the QuantLib documentation
for details on the QuantLib implementation.

### Value

None, but side effects of displaying content.

### Note

The interface might change in future release as QuantLib stabilises its own API.

### Author(s)

Khanh Nguyen <knguyen@cs.umb.edu>; Dirk Eddelbuettel <edd@debian.org> for the R
interface; the QuantLib Group for QuantLib

### References

http://quantlib.org for details on QuantLib.

**Examples**

```
## This data is taken from sample code shipped with QuantLib 0.9.7
## from the file Examples/Swap/swapvaluation
params <- list(tradeDate=as.Date('2004-09-20'),
               settleDate=as.Date('2004-09-22'),
               dt=.25,
               interpWhat="discount",
               interpHow="loglinear")

## We got numerical issues for the spline interpolation if we add
## any on of these three extra futures, at least with QuantLib 0.9.7
## The curve data comes from QuantLib's Examples/Swap/swapvaluation.cpp
tsQuotes <- list(d1w = 0.0382,
                 d1m = 0.0372,
                 fut1=96.2875,
                 fut2=96.7875,
                 fut3=96.9875,
                 fut4=96.6875,
                 fut5=96.4875,
                 fut6=96.3875,
                 fut7=96.2875,
                 fut8=96.0875,
                 s2y = 0.037125,
                 s3y = 0.0398,
                 s5y = 0.0443,
                 s10y = 0.05165,
                 s15y = 0.055175)

times <- seq(0,10,.1)

discountCurve <- DiscountCurve(params, tsQuotes, times)

# price a zero coupon bond
bondparams <- list(faceAmount=100, issueDate=as.Date("2004-11-30"),
              maturityDate=as.Date("2008-11-30"), redemption=100 )
dateparams <-list(settlementDays=1, calendar="us", businessDayConvention=4)
ZeroCouponBond(bondparams, discountCurve, dateparams)

# price a fixed rate coupon bond

bondparams <- list(faceAmount=100, issueDate=as.Date("2004-11-30"),
              maturityDate=as.Date("2008-11-30"), redemption=100,
              effectiveDate=as.Date("2004-11-30"))
dateparams <- list(settlementDays=1, calendar="us", dayCounter = 1, period=3,
                   businessDayConvention = 4, terminationDateConvention=4,
                   dateGeneration=1, endOfMonth=1)
rates <- c(0.02875)
FixedRateBond(bondparams, rates, discountCurve, dateparams)

# price a floating rate bond
bondparams <- list(faceAmount=100, issueDate=as.Date("2004-11-30"),
```

```
                   maturityDate=as.Date("2008-11-30"), redemption=100,
                   effectiveDate=as.Date("2004-11-30"))

dateparams <- list(settlementDays=1, calendar="us", dayCounter = 1, period=3,
                   businessDayConvention = 1, terminationDateConvention=1,
                   dateGeneration=0, endOfMonth=0, fixingDays = 1)

gearings <- c()
spreads <- c()
caps <- c()
floors <- c()

iborCurve <- DiscountCurve(params,list(flat=0.05), times)
ibor <- list(type="USDLibor", length=6, inTermOf="Month",
             term=iborCurve)
FloatingRateBond(bondparams, gearings, spreads, caps, floors,
                 ibor, discountCurve, dateparams)
```

---

BondUtilities       *Bond parameter conversion utilities*

---

### Description

These functions are using internally to convert from the characters at the R level to the enum types
used at the C++ level. They are documented here mostly to provide a means to look up some of the
possible values—the user is not expected to call these functions directly..

### Usage

```
matchBDC(bdc = c("Following", "ModifiedFollowing", "Preceding", "ModifiedPreceding"
matchCompounding(cp = c("Simple", "Compounded", "Continuous", "SimpleThenCompounded
matchDayCounter(daycounter = c("Actual360", "ActualFixed", "ActualActual", "Busines
matchDateGen(dg = c("Backward", "Forward", "Zero", "ThirdWednesday", "Twentieth", "
matchFrequency(freq = c("NoFrequency","Once", "Annual", "Semiannual", "EveryFourthM
matchParams(params)
```

### Arguments

| | |
|---|---|
| bdc | A string identifying one of the possible business day convention values. |
| cp | A string identifying one of the possible compounding frequency values. |
| daycounter | A string identifying one of the possible day counter scheme values. |
| dg | A string identifying one of the possible date generation scheme values. |
| freq | A string identifying one of the possible (dividend) frequency values. |
| params | A named vector containing the other parameters as components. |

### Details

The QuantLib documentation should be consulted for details.

**Value**

Each function converts the given character value into a corresponding numeric entry. For `matchParams`, an named vector of strings is converted into a named vector of numerics..

**Note**

The interface might change in future release as `QuantLib` stabilises its own API.

**Author(s)**

Khanh Nguyen `<knguyen@cs.umb.edu>` for the R interface; the QuantLib Group for `QuantLib`

**References**

`http://quantlib.org` for details on `QuantLib`.

---

businessDaysBetween

*Calendar functions from QuantLib*

---

**Description**

The `businessDaysBetween` function evaluates two given dates in the context of the given calendar, and returns a vector that gives the number of business day between.

**Usage**

```
businessDaysBetween(calendar="TARGET", from=Sys.Date(),
to = Sys.Date() + 5, includeFirst = 1, includeLast = 0)
```

**Arguments**

| | |
|---|---|
| `calendar` | A string identifying one of the supported QuantLib calendars, see Details for more |
| `from` | A vector (or scalar) of `Date` types. |
| `to` | A vector (or scalar) of `Date` types. |
| `includeFirst` | boolean that indicates whether the calculation should include the first day. Default = true |
| `includeLast` | Default = false |

**Details**

The calendars are coming from QuantLib, and the QuantLib documentation should be consulted for details.

Currently, the following strings are recognised: TARGET (a default calendar), Canada and Canada/Settlement, Canada/TSX, Germany and Germany/FrankfurtStockExchange, Germany/Settlement, Germany/Xetra, Germany/Eurex, Italy and Italy/Settlement, Italy/Exchange, Japan, UnitedKingdom and UnitedKingdom/Settlement, UnitedKingdom/Exchange, UnitedKingdom/Metals, UnitedStates and UnitedStates/Settlement, UnitedStates/NYSE, UnitedStates/GovernmentBond, UnitedStates/NERC.

(In case of multiples entries per country, the country default is listed right after the country itself. Using the shorter form is equivalent.)

**Value**

An named vector of number.

**Note**

The interface might change in future release as `QuantLib` stabilises its own API.

**Author(s)**

Dirk Eddelbuettel `<edd@debian.org>` for the R interface; Khanh Nguyen `<nguyen.h.khanh@gmail.com>` for the implementation; the QuantLib Group for `QuantLib`

**References**

`http://quantlib.org` for details on `QuantLib`.

**Examples**

```
  from <- as.Date("2009-04-07")
to<-as.Date("2009-04-14")
businessDaysBetween("UnitedStates", from, to)
```

---

Calendars                *Calendar functions from QuantLib*

---

**Description**

The `businessDay` function evaluates the given dates in the context of the given calendar, and returns a vector of booleans indicating business day status.

**Usage**

```
businessDay(calendar="TARGET", dates=Sys.Date())
```

### Arguments

calendar      A string identifying one of the supported QuantLib calendars, see Details for more

dates         A vector (or scalar) of Date types.

### Details

The calendars are coming from QuantLib, and the QuantLib documentation should be consulted for details.

Currently, the following strings are recognised: TARGET (a default calendar), Canada and Canada/Settlement, Canada/TSX, Germany and Germany/FrankfurtStockExchange, Germany/Settlement, Germany/Xetra, Germany/Eurex, Italy and Italy/Settlement, Italy/Exchange, Japan, UnitedKingdom and United-Kingdom/Settlement, UnitedKingdom/Exchange, UnitedKingdom/Metals, UnitedStates and UnitedStates/Settlement, UnitedStates/NYSE, UnitedStates/GovernmentBond, UnitedStates/NERC.

(In case of multiples entries per country, the country default is listed right after the country itself. Using the shorter form is equivalent.)

### Value

An named vector of booleans each of which is true if the corresponding date is a business day in the given calendar. The element names are the dates (formatted as text in yyyy-mm-dd format).

### Note

The interface might change in future release as QuantLib stabilises its own API.

### Author(s)

Dirk Eddelbuettel <edd@debian.org> for the R interface; the QuantLib Group for QuantLib

### References

http://quantlib.org for details on QuantLib.

### Examples

```
dates <- seq(from=as.Date("2009-04-07"), to=as.Date("2009-04-14"), by=1)
businessDay("UnitedStates", dates)
businessDay("UnitedStates/Settlement", dates)      ## same as previous
businessDay("UnitedStates/NYSE", dates)            ## stocks
businessDay("UnitedStates/GovernmentBond", dates)  ## bonds
businessDay("UnitedStates/NERC", dates)            ## energy
```

---

CallableBond          *CallableBond evaluation*

---

**Description**

The `CallableBond` function sets up and evaluates a callable fixed rate bond using Hull-White model and a TreeCallableFixedBondEngine pricing engine. For more detail, see the source codes in quantlib's example folder, Examples/CallableBond/CallableBond.cpp

**Usage**

```
## Default S3 method:
CallableBond(bondparams, hullWhite, coupon, dateparams)
## S3 method for class 'Bond':
plot
## S3 method for class 'Bond':
print
## S3 method for class 'Bond':
summary
```

**Arguments**

bondparams      a named list whose elements are:

| | |
|---|---|
| issueDate | a Date, the bond's issue date |
| maturityDate | a Date, the bond's maturity date |
| faceAmount | (Optional) a double, face amount of the bond. Default value is 100. |
| redemption | (Optional) a double, percentage of the initial face amount that will be returned at maturity date. Default value is 100. |
| callSch | (Optional) a data frame whose columns are "Price", "Type" and "Date" corresponding to QuantLib's CallabilitySchedule. Defaule is an empty frame, or no callability. |

hullWhite     a named list whose elements are parameters needed to set up a HullWhite pricing engine in QuantLib:

| | |
|---|---|
| term | a double, to set up a flat rate yield term structure |
| alpha | a double, Hull-White model's alpha value |
| sigma | a double, Hull-White model's sigma value |
| gridIntervals. | a double, time intervals parameter to set up the TreeCallableFixedBondEngine |

Currently, the codes only support a flat rate yield term structure. For more detail,

see QuantLib's doc on HullWhite and TreeCallableFixedBondEngine.

coupon          a numeric vector of coupon rates

dateparams      (Optional) a named list, QuantLib's date parameters of the bond.

    settlementDays                (Optional) a double, settlement days.
Default value is 1.

| | |
|---|---|
| settlementDays | (Optional) a double, settlement days. Default value is 1. |
| calendar | (Optional) a string, either 'us' or 'uk' corresponding to US Goverment Bond calendar and UK Exchange calendar. Default value is 'us'. |
| dayCounter | (Optional) a number or string, day counter convention. See Enum. Default value is 'Thirty360' |
| period | (Optional) a number or string, interest compounding interval. See Enum. Default value is 'Semiannual'. |
| businessDayConvention | (Optional) a number or string, business day convention. See Enum. Default value is 'Following'. |
| terminationDateConvention | (Optional) a number or string termination day convention. See Enum. Default value is'Following'. |

See example below.

## Details

Please see any decent Finance textbook for background reading, and the `QuantLib` documentation for details on the `QuantLib` implementation.

## Value

The `CallableBond` function returns an object of class `CallableBond` (which inherits from class `Bond`). It contains a list with the following components:

| | |
|---|---|
| NPV | net present value of the bond |
| cleanPrice | price price of the bond |
| dirtyPrice | dirty price of the bond |
| accruedAmount | |
| | accrued amount of the bond |
| yield | yield of the bond |
| cashFlows | cash flows of the bond |

## Note

The interface might change in future release as `QuantLib` stabilises its own API.

## Author(s)

Khanh Nguyen <knguyen@cs.umb.edu> for the inplementation; Dirk Eddelbuettel <edd@debian.org>
for the R interface; the QuantLib Group for QuantLib

## References

<http://quantlib.org> for details on QuantLib.

## Examples

```
#set-up a HullWhite according to example from QuantLib
HullWhite <- list(term = 0.055, alpha = 0.03, sigma = 0.01,
                  gridIntervals = 40)

#callability schedule dataframe
Price <- rep(as.double(100),24)
Type <- rep(as.character("C"), 24)
Date <- seq(as.Date("2006-09-15"), by = '3 months', length = 24)
callSch <- data.frame(Price, Type, Date)
callSch$Type <- as.character(callSch$Type)

bondparams <- list(faceAmount=100, issueDate = as.Date("2004-09-16"),
                   maturityDate=as.Date("2012-09-16"), redemption=100,
                   callSch = callSch)
dateparams <- list(settlementDays=3, calendar="us",
                   dayCounter = "ActualActual",
                   period="Quarterly",
                   businessDayConvention = "Unadjusted",
                   terminationDateConvention= "Unadjusted")
coupon <- c(0.0465)

CallableBond(bondparams, HullWhite, coupon, dateparams)
#examples using default values
CallableBond(bondparams, HullWhite, coupon)
dateparams <- list(
                   period="Quarterly",
                   businessDayConvention = "Unadjusted",
                   terminationDateConvention= "Unadjusted")
CallableBond(bondparams, HullWhite, coupon, dateparams)


bondparams <- list(issueDate = as.Date("2004-09-16"),
                   maturityDate=as.Date("2012-09-16")
                   )
CallableBond(bondparams, HullWhite, coupon, dateparams)
```

---

ConvertibleFixedCouponBond

*Convertible Fixed Coupon Bond evaluation*

---

**Description**

The `ConvertibleFixedCouponBond` function setups and evaluates a ConvertibleFixedCoupon-
Bond using QuantLib's BinomialConvertibleEngine [http://quantlib.org/reference/](http://quantlib.org/reference/class_quant_lib_1_1_binomial_convertible_engine.html)
[class_quant_lib_1_1_binomial_convertible_engine.html](http://quantlib.org/reference/class_quant_lib_1_1_binomial_convertible_engine.html) and BlackScholesMer-
tonProcess [http://quantlib.org/reference/class_quant_lib_1_1_black_scholes_](http://quantlib.org/reference/class_quant_lib_1_1_black_scholes_merton_process.html)
[merton_process.html](http://quantlib.org/reference/class_quant_lib_1_1_black_scholes_merton_process.html).The NPV, clean price, dirty price, accrued interest, yield and cash
flows of the bond is returned. For detail, see test-suite/convertiblebond.cpp

**Usage**

```
## Default S3 method:
ConvertibleFixedCouponBond(bondparams, coupon, process, dateparams)
## S3 method for class 'Bond':
plot
## S3 method for class 'Bond':
print
## S3 method for class 'Bond':
summary
```

**Arguments**

bondparams     bond parameters, a named list whose elements are:

| | |
|---|---|
| issueDate | a Date, the bond's issue date |
| maturityDate | a Date, the bond's maturity date |
| creditSpread | a double, credit spread parameter in the constructor of the bond. |
| conversitionRatio | a double, conversition ratio parameter in the constructor of the bond. |
| exercise | (Optional) a string, either "eu" for European option, or "am" for American option. Default value is 'am'. |
| faceAmount | (Optional) a double, face amount of the bond. Default value is 100. |
| redemption | (Optional) a double, percentage of the initial face amount that will be returned at maturity date. Default value is 100. |
| divSch | (Optional) a data frame whose columns are "Type", "Amount", "Rate", and "Date" corresponding to QuantLib's DividendSchedule. Default value is an empty frame, or no dividend. |
| callSch | (Optional) a data frame whose columns are "Price", "Type" and "Date" corresponding to QuantLib's CallabilitySchedule. Defaule is an empty frame, or no callability. |

coupon          a double vector of coupon rate

process          arguments to construct a BlackScholes process and set up the binomial pricing
                 engine for this bond.

| | |
|---|---|
| underlying | a double, flat underlying term structure |
| volatility | a double, flat volatility term structure |
| dividendYield | a DiscountCurve object |
| riskFreeRate | a DiscountCurve object |

dateparams     (Optional) a named list, QuantLib's date parameters of the bond.

| | |
|---|---|
| settlementDays | (Optional) a double, settlement days. Default value is 1. |
| calendar | (Optional) a string, either 'us' or 'uk' corresponding to US Goverment Bond calendar and UK Exchange calendar. Default value is 'us'. |
| dayCounter | (Optional) a number or string, day counter convention. See [Enum](). Default value is 'Thirty360' |
| period | (Optional) a number or string, interest compounding interval. See [Enum](). Default value is 'Semiannual'. |
| businessDayConvention | (Optional) a number or string, business day convention. See [Enum](). Default value is 'Following'. |

                 See example below.

### Details

Please see any decent Finance textbook for background reading, and the QuantLib documentation
for details on the QuantLib implementation.

### Value

The ConvertibleFixedCouponBond function returns an object of class ConvertibleFixedCouponBond
(which inherits from class Bond). It contains a list with the following components:

NPV            net present value of the bond

cleanPrice     price price of the bond

dirtyPrice     dirty price of the bond

accruedAmount

               accrued amount of the bond

yield          yield of the bond

cashFlows      cash flows of the bond

**Author(s)**

Khanh Nguyen <knguyen@cs.umb.edu> for the inplementation; Dirk Eddelbuettel <edd@debian.org> for the R interface; the QuantLib Group for QuantLib

**References**

http://quantlib.org/reference/class_quant_lib_1_1_convertible_zero_coupon_bond.html

**Examples**

```
#this follow an example in test-suite/convertiblebond.cpp
#for ConvertibleFixedCouponBond

#set up arguments to build a pricing engine.
params <- list(tradeDate=Sys.Date()-2,
               settleDate=Sys.Date(),
               dt=.25,
               interpWhat="discount",
               interpHow="loglinear")
times <- seq(0,10,.1)

dividendYield <- DiscountCurve(params, list(flat=0.02), times)
riskFreeRate <- DiscountCurve(params, list(flat=0.05), times)

dividendSchedule <- data.frame(Type=character(0), Amount=numeric(0),
                          Rate = numeric(0), Date = as.Date(character(0)))
callabilitySchedule <- data.frame(Price = numeric(0), Type=character(0),
                          Date = as.Date(character(0)))

process <- list(underlying=50, divYield = dividendYield,
                rff = riskFreeRate, volatility=0.15)

today <- Sys.Date()
bondparams <- list(exercise="am", faceAmount=100, divSch = dividendSchedule,
                   callSch = callabilitySchedule, redemption=100,
                   creditSpread=0.005, conversionRatio = 0.0000000001,
                   issueDate=as.Date(today+2),
                   maturityDate=as.Date(today+3650))
dateparams <- list(settlementDays=3,
                   dayCounter="Actual360",
                   period = "Once", calendar = "us",
                   businessDayConvention="Following"
                   )
coupon <- c(0.05)
ConvertibleFixedCouponBond(bondparams, coupon, process, dateparams)

#example with default value
ConvertibleFixedCouponBond(bondparams, coupon, process)

dateparams <- list(settlementDays=3,
                   dayCounter="Actual360")
```

```
ConvertibleFixedCouponBond(bondparams, coupon, process, dateparams)

bondparams <- list(creditSpread=0.005, conversionRatio = 0.0000000001,
                   issueDate=as.Date(today+2),
                   maturityDate=as.Date(today+3650))
ConvertibleFixedCouponBond(bondparams, coupon, process, dateparams)
```

---

ConvertibleFloatingCouponBond

*Convertible Floating Coupon Bond evaluation*

---

### Description

The `ConvertibleFloatingCouponBond` function setups and evaluates a ConvertibleFixed-CouponBond using QuantLib's BinomialConvertibleEngine [http://quantlib.org/reference/](http://quantlib.org/reference/) [class_quant_lib_1_1_binomial_convertible_engine.html](class_quant_lib_1_1_binomial_convertible_engine.html) and BlackScholesMertonProcess [http://quantlib.org/reference/class_quant_lib_1_1_black_scholes_](http://quantlib.org/reference/class_quant_lib_1_1_black_scholes_) [merton_process.html](merton_process.html). The NPV, clean price, dirty price, accrued interest, yield and cash flows of the bond is returned. For detail, see test-suite/convertiblebond.cpp

### Usage

```
## Default S3 method:
ConvertibleFloatingCouponBond(bondparams, iborindex,
                                        spread, process, dateparams)
## S3 method for class 'Bond':
plot
## S3 method for class 'Bond':
print
## S3 method for class 'Bond':
summary
```

### Arguments

bondparams      bond parameters, a named list whose elements are:

| | |
|---|---|
| issueDate | a Date, the bond's issue date |
| maturityDate | a Date, the bond's maturity date |
| creditSpread | a double, credit spread parameter in the constructor of the bond. |
| conversitionRatio | a double, conversition ratio parameter in the constructor of the bond. |
| exercise | (Optional) a string, either "eu" for European option, or "am" for American option. Default value is 'am'. |
| faceAmount | (Optional) a double, face amount of the bond. |

| | |
|---|---|
| | Default value is 100. |
| redemption | (Optional) a double, percentage of the initial face amount that will be returned at maturity date. Default value is 100. |
| divSch | (Optional) a data frame whose columns are "Type", "Amount", "Rate", and "Date" corresponding to QuantLib's DividendSchedule. Default value is an empty frame, or no dividend. |
| callSch | (Optional) a data frame whose columns are "Price", "Type" and "Date" corresponding to QuantLib's CallabilitySchedule. Defaule is an empty frame, or no callability. |

| | |
|---|---|
| iborindex | a DiscountCurve object, represents an IborIndex |
| spread | a double vector, represents paramter 'spreads' in ConvertibleFloatingBond's constructor. |
| process | arguments to construct a BlackScholes process and set up the binomial pricing engine for this bond. |

| | |
|---|---|
| underlying | a double, flat underlying term structure |
| volatility | a double, flat volatility term structure |
| dividendYield | a DiscountCurve object |
| riskFreeRate | a DiscountCurve object |

| | |
|---|---|
| dateparams | (Optional) a named list, QuantLib's date parameters of the bond. |

| | |
|---|---|
| settlementDays | (Optional) a double, settlement days. Default value is 1. |
| calendar | (Optional) a string, either 'us' or 'uk' corresponding to US Goverment Bond calendar and UK Exchange calendar. Default value is 'us'. |
| dayCounter | (Optional) a number or string, day counter convention. See [Enum]. Default value is 'Thirty360' |
| period | (Optional) a number or string, interest compounding interval. See [Enum]. Default value is 'Semiannual'. |
| businessDayConvention | (Optional) a number or string, business day convention. See [Enum]. Default value is 'Following'. |

See example below.

**Details**

Please see any decent Finance textbook for background reading, and the `QuantLib` documentation for details on the `QuantLib` implementation.

**Value**

The `ConvertibleFloatingCouponBond` function returns an object of class `ConvertibleFloatingCouponBond` (which inherits from class `Bond`). It contains a list with the following components:

| | |
|---|---|
| NPV | net present value of the bond |
| cleanPrice | price price of the bond |
| dirtyPrice | dirty price of the bond |
| accruedAmount | |
| | accrued amount of the bond |
| yield | yield of the bond |
| cashFlows | cash flows of the bond |

**Author(s)**

Khanh Nguyen <knguyen@cs.umb.edu> for the inplementation; Dirk Eddelbuettel <edd@debian.org> for the R interface; the QuantLib Group for `QuantLib`

**References**

http://quantlib.org/reference/class_quant_lib_1_1_convertible_zero_
coupon_bond.html

**Examples**

```
#this follow an example in test-suite/convertiblebond.cpp
params <- list(tradeDate=Sys.Date()-2,
               settleDate=Sys.Date(),
               dt=.25,
               interpWhat="discount",
               interpHow="loglinear")
times <- seq(0,10,.1)


dividendYield <- DiscountCurve(params, list(flat=0.02), times)
riskFreeRate <- DiscountCurve(params, list(flat=0.05), times)

dividendSchedule <- data.frame(Type=character(0), Amount=numeric(0),
                          Rate = numeric(0), Date = as.Date(character(0)))
callabilitySchedule <- data.frame(Price = numeric(0), Type=character(0),
                          Date = as.Date(character(0)))

process <- list(underlying=50, divYield = dividendYield,
                rff = riskFreeRate, volatility=0.15)
```

```
today <- Sys.Date()
bondparams <- list(exercise="am", faceAmount=100, divSch = dividendSchedule,
                   callSch = callabilitySchedule, redemption=100,
                   creditSpread=0.005, conversionRatio = 0.0000000001,
                   issueDate=as.Date(today+2),
                   maturityDate=as.Date(today+3650))
dateparams <- list(settlementDays=3,
                   dayCounter="Actual360",
                   period = "Once", calendar = "us",
                   businessDayConvention="Following"
                   )

lengths <- c(2,4,6,8,10,12,14,16,18,20,22,24,26,28,30)
coupons <- c( 0.0200, 0.0225, 0.0250, 0.0275, 0.0300,
              0.0325, 0.0350, 0.0375, 0.0400, 0.0425,
              0.0450, 0.0475, 0.0500, 0.0525, 0.0550 )
curvedateparams <- list(settlementDays=0, period="Annual",
                   dayCounter="SimpleDayCounter",
                   businessDayConvention ="Unadjusted")
curveparams <- list(method="ExponentialSplinesFitting",
                    origDate = Sys.Date())
curve <- FittedBondCurve(curveparams, lengths, coupons, curvedateparams)
iborindex <- list(type="USDLibor", length=6,
                  inTermOf="Month", term=curve)
spreads <- c()
ConvertibleFloatingCouponBond(bondparams, iborindex,spreads, process, dateparams)


#example using default values
ConvertibleFloatingCouponBond(bondparams, iborindex,spreads, process)

dateparams <- list(settlementDays=3,
                   period = "Once",
                   businessDayConvention="Unadjusted"
                   )

bondparams <- list(
                   creditSpread=0.005, conversionRatio = 0.0000000001,
                   issueDate=as.Date(today+2),
                   maturityDate=as.Date(today+3650))
ConvertibleFloatingCouponBond(bondparams, iborindex,
spreads, process, dateparams)
```

---

ConvertibleZeroCouponBond

*Convertible Zero Coupon Bond evaluation*

---

**Description**

The `ConvertibleZeroCouponBond` function setups and evaluates a ConvertibleFixedCoupon-
Bond using QuantLib's BinomialConvertibleEngine [http://quantlib.org/reference/](http://quantlib.org/reference/class_quant_lib_1_1_binomial_convertible_engine.html)
[class_quant_lib_1_1_binomial_convertible_engine.html](http://quantlib.org/reference/class_quant_lib_1_1_binomial_convertible_engine.html) and BlackScholesMer-
tonProcess [http://quantlib.org/reference/class_quant_lib_1_1_black_scholes_](http://quantlib.org/reference/class_quant_lib_1_1_black_scholes_merton_process.html)
[merton_process.html](http://quantlib.org/reference/class_quant_lib_1_1_black_scholes_merton_process.html). The NPV, clean price, dirty price, accrued interest, yield and cash
flows of the bond is returned. For detail, see test-suite/convertiblebond.cpp.

**Usage**

```
## Default S3 method:
ConvertibleZeroCouponBond(bondparams, process, dateparams)
## S3 method for class 'Bond':
plot
## S3 method for class 'Bond':
print
## S3 method for class 'Bond':
summary
```

**Arguments**

bondparams      bond parameters, a named list whose elements are:

| | |
|---|---|
| issueDate | a Date, the bond's issue date |
| maturityDate | a Date, the bond's maturity date |
| creditSpread | a double, credit spread parameter in the constructor of the bond. |
| conversitionRatio | a double, conversition ratio parameter in the constructor of the bond. |
| exercise | (Optional) a string, either "eu" for European option, or "am" for American option. Default value is 'am'. |
| faceAmount | (Optional) a double, face amount of the bond. Default value is 100. |
| redemption | (Optional) a double, percentage of the initial face amount that will be returned at maturity date. Default value is 100. |
| divSch | (Optional) a data frame whose columns are "Type", "Amount", "Rate", and "Date" corresponding to QuantLib's DividendSchedule. Default value is an empty frame, or no dividend. |
| callSch | (Optional) a data frame whose columns are "Price", "Type" and "Date" corresponding to QuantLib's CallabilitySchedule. Defaule is an empty frame, or no callability. |

process      arguments to construct a BlackScholes process and set up the binomial pricing

engine for this bond.

| underlying | a double, flat underlying term structure |
| volatility | a double, flat volatility term structure |
| dividendYield | a DiscountCurve object |
| riskFreeRate | a DiscountCurve object |

dateparams    (Optional) a named list, QuantLib's date parameters of the bond.

| settlementDays | (Optional) a double, settlement days. Default value is 1. |
| calendar | (Optional) a string, either 'us' or 'uk' corresponding to US Goverment Bond calendar and UK Exchange calendar. Default value is 'us'. |
| dayCounter | (Optional) a number or string, day counter convention. See Enum. Default value is 'Thirty360' |
| period | (Optional) a number or string, interest compounding interval. See Enum. Default value is 'Semiannual'. |
| businessDayConvention | (Optional) a number or string, business day convention. See Enum. Default value is 'Following'. |

See example below.

**Details**

Please see any decent Finance textbook for background reading, and the QuantLib documentation for details on the QuantLib implementation.

**Value**

The ConvertibleZeroCouponBond function returns an object of class ConvertibleZeroCouponBond (which inherits from class Bond). It contains a list with the following components:

NPV            net present value of the bond

cleanPrice     price price of the bond

dirtyPrice     dirty price of the bond

accruedAmount
               accrued amount of the bond

yield          yield of the bond

cashFlows      cash flows of the bond

**Author(s)**

Khanh Nguyen <knguyen@cs.umb.edu> for the inplementation; Dirk Eddelbuettel <edd@debian.org> for the R interface; the QuantLib Group for QuantLib

**References**

http://quantlib.org/reference/class_quant_lib_1_1_convertible_zero_coupon_bond.html

**Examples**

```
#this follow an example in test-suite/convertiblebond.cpp
params <- list(tradeDate=Sys.Date()-2,
               settleDate=Sys.Date(),
               dt=.25,
               interpWhat="discount",
               interpHow="loglinear")
times <- seq(0,10,.1)


dividendYield <- DiscountCurve(params, list(flat=0.02), times)
riskFreeRate <- DiscountCurve(params, list(flat=0.05), times)

dividendSchedule <- data.frame(Type=character(0), Amount=numeric(0),
                          Rate = numeric(0), Date = as.Date(character(0)))
callabilitySchedule <- data.frame(Price = numeric(0), Type=character(0),
                          Date = as.Date(character(0)))

process <- list(underlying=50, divYield = dividendYield,
                rff = riskFreeRate, volatility=0.15)

today <- Sys.Date()
bondparams <- list(exercise="am", faceAmount=100, divSch = dividendSchedule,
                   callSch = callabilitySchedule, redemption=100,
                   creditSpread=0.005, conversionRatio = 0.0000000001,
                   issueDate=as.Date(today+2),
                   maturityDate=as.Date(today+3650))
dateparams <- list(settlementDays=3,
                   dayCounter="Actual360",
                   period = "Once", calendar = "us",
                   businessDayConvention="Following"
                   )

ConvertibleZeroCouponBond(bondparams, process, dateparams)

#example with default values
ConvertibleZeroCouponBond(bondparams, process)


bondparams <- list(creditSpread=0.005,
                   conversionRatio=0.0000000001,
                   issueDate=as.Date(today+2),
```

```
                       maturityDate=as.Date(today+3650))

dateparams <- list(settlementDays=3, dayCounter='Actual360')
ConvertibleZeroCouponBond(bondparams, process, dateparams)
ConvertibleZeroCouponBond(bondparams, process)
```

---

dayCount                    *DayCounter functions from QuantLib*

---

### Description

The dayCount function returns the number of day between two dates given a day counter Enum

### Usage

```
dayCount(startDates, endDates, dayCounters)
```

### Arguments

startDates    A vector of Date type.

endDates      A vector of Date type.

dayCounters   A vector of numeric type. See Enum

### Details

The day counters are coming from QuantLib, and the QuantLib documentation should be consulted for details. See Enum and http://quantlib.org/reference/class_quant_lib_1_1_day_counter.html

### Value

A numeric vector contains the number of day between two dates from the input.

### Note

The interface might change in future release as QuantLib stabilises its own API.

### Author(s)

Dirk Eddelbuettel <edd@debian.org> for the R interface; Khanh Nguyen <nguyen.h.khanh@gmail.com> for the implementation; the QuantLib Group for QuantLib

### References

http://quantlib.org for details on QuantLib.

**Examples**

```
startDates <- seq(from=as.Date("2009-04-07"), to=as.Date("2009-04-14"),by=1)
endDates <- seq(from=as.Date("2009-11-07"), to=as.Date("2009-11-14"), by=1)
dayCounters <- c(0,1,2,3,4,5,6,1)
dayCount(startDates, endDates, dayCounters)
```

---

| DiscountCurve | *Returns the discount curve (with zero rates and forwards) given times* |
|---|---|

---

**Description**

DiscountCurve constructs the spot term structure of interest rates based on input market data including the settlement date, deposit rates, futures prices, FRA rates, or swap rates, in various combinations. It returns the corresponding discount factors, zero rates, and forward rates for a vector of times that is specified as input.

**Usage**

```
DiscountCurve(params, tsQuotes, times)
```

**Arguments**

params        A list specifying the tradeDate (month/day/year), settleDate, forward
              rate time span dt, and two curve construction options: interpWhat (with
              possible values discount, forward, and zero) and interpHow (with
              possible values linear, loglinear, and spline). spline here means
              cubic spline interpolation of the interpWhat value.

tsQuotes      Market quotes used to construct the spot term structure of interest rates. Must
              be a list of name/value pairs, where the currently recognized names are:

| | |
|---|---|
| flat | rate for a flat yield curve |
| d1w | 1-week deposit rate |
| d1m | 1-month deposit rate |
| d3m | 3-month deposit rate |
| d6m | 6-month deposit rate |
| d9m | 9-month deposit rate |
| d1y | 1-year deposit rate |
| s2y | 2-year swap rate |
| s3y | 3-year swap rate |
| s5y | 5-year swap rate |
| s10y | 10-year swap rate |
| s15y | 15-year swap rate |
| s20y | 20-year swap rate |
| s30y | 30-year swap rate |
| fut1−fut8 | 3-month futures contracts |
| fra3x6 | 3x6 FRA |
| fra6x9 | 6x9 FRA |
| fra6x12 | 6x12 FRA |

Here rates are expected as fractions (so 5% means .05). If `flat` is specified it must be the first and only item in the list. The eight futures correspond to the first eight IMM dates. The maturity dates of the instruments specified need not be ordered, but they must be distinct.

times   A vector of times at which to return the discount factors, forward rates, and zero rates. Times must be specified such that the largest time plus `dt` does not exceed the longest maturity of the instruments used for calibration (no extrapolation).

### Details

This function is based on `QuantLib` Version 0.3.10. It introduces support for fixed-income instruments in `RQuantLib`.

Forward rates and zero rates are computed assuming continuous compounding, so the forward rate $f$ over the period from $t_1$ to $t_2$ is determined by the relation

$$d_1/d_2 = e^{f(t_2 - t_1)},$$

where $d_1$ and $d_2$ are discount factors corresponding to the two times. In the case of the zero rate $t_1$ is the current time (the spot date).

Curve construction can be a delicate problem and the algorithms may fail for some input data sets and/or some combinations of the values for `interpWhat` and `interpHow`. Fortunately, the C++ exception mechanism seems to work well with the R interface, and `QuantLib` exceptions are propagated back to the R user, usually with a message that indicates what went wrong. (The first part of the message contains technical information about the precise location of the problem in the `QuantLib` code. Scroll to the end to find information that is meaningful to the R user.)

### Value

`DiscountCurve` returns a list containing:

times       Vector of input times
discounts    Corresponding discount factors
forwards     Corresponding forward rates with time span `dt`
zerorates    Corresponding zero coupon rates
flatQuotes   True if a flat quote was used, False otherwise
params       The input parameter list

### Author(s)

Dominick Samperi

### References

Brigo, D. and Mercurio, F. (2001) *Interest Rate Models: Theory and Practice*, Springer-Verlag, New York.

For information about `QuantLib` see http://quantlib.org.

For information about `RQuantLib` see http://dirk.eddelbuettel.com/code/rquantlib.html.

**See Also**

BermudanSwaption

**Examples**

```
savepar <- par(mfrow=c(3,3), mar=c(4,4,2,0.5))

## This data is taken from sample code shipped with QuantLib 0.9.7
## from the file Examples/Swap/swapvaluation
params <- list(tradeDate=as.Date('2004-09-20'),
               settleDate=as.Date('2004-09-22'),
               dt=.25,
               interpWhat="discount",
               interpHow="loglinear")

## We get numerical issue for the spline interpolation if we add
## any on of these three extra futures
tsQuotes <- list(d1w = 0.0382,
                 d1m = 0.0372,
                 d3m = 0.0363,
                 d6m = 0.0353,
                 d9m = 0.0348,
                 d1y = 0.0345,
#                 fut1=96.2875,
                 fut2=96.7875,
                 fut3=96.9875,
                 fut4=96.6875,
                 fut5=96.4875,
#                 fut6=96.3875,
                 fut7=96.2875,
#                 fut8=96.0875,
                 s2y = 0.037125,
                 s3y = 0.0398,
                 s5y = 0.0443,
                 s10y = 0.05165,
                 s15y = 0.055175)

times <- seq(0,10,.1)

# Loglinear interpolation of discount factors
curves <- DiscountCurve(params, tsQuotes, times)
plot(curves,setpar=FALSE)

# Linear interpolation of discount factors
params$interpHow="linear"
curves <- DiscountCurve(params, tsQuotes, times)
plot(curves,setpar=FALSE)

# Spline interpolation of discount factors
params$interpHow="spline"
curves <- DiscountCurve(params, tsQuotes, times)
```

```
plot(curves,setpar=FALSE)

par(savepar)
```

---

endOfMonth                *Calendar functions from QuantLib*

---

### Description

The `endOfMonth` function evaluates the given dates in the context of the given calendar, and returns a vector that corresponds to the end of month.

### Usage

```
endOfMonth(calendar="TARGET", dates=Sys.Date())
```

### Arguments

calendar     A string identifying one of the supported QuantLib calendars, see Details for more
dates        A vector (or scalar) of `Date` types.

### Details

The calendars are coming from QuantLib, and the QuantLib documentation should be consulted for details.

Currently, the following strings are recognised: TARGET (a default calendar), Canada and Canada/Settlement, Canada/TSX, Germany and Germany/FrankfurtStockExchange, Germany/Settlement, Germany/Xetra, Germany/Eurex, Italy and Italy/Settlement, Italy/Exchange, Japan, UnitedKingdom and United-Kingdom/Settlement, UnitedKingdom/Exchange, UnitedKingdom/Metals, UnitedStates and United-States/Settlement, UnitedStates/NYSE, UnitedStates/GovernmentBond, UnitedStates/NERC.

(In case of multiples entries per country, the country default is listed right after the country itself. Using the shorter form is equivalent.)

### Value

An named vector of dates, each of which is the end of month date that corresponds to the input dates. The element names are the dates (formatted as text in yyyy-mm-dd format).

### Note

The interface might change in future release as `QuantLib` stabilises its own API.

### Author(s)

Dirk Eddelbuettel <edd@debian.org> for the R interface; Khanh Nguyen <nguyen.h.khanh@gmail.com> for the implementation; the QuantLib Group for `QuantLib`

## References

<http://quantlib.org> for details on `QuantLib`.

## Examples

```
dates <- seq(from=as.Date("2009-04-07"), to=as.Date("2009-04-14"), by=1)
endOfMonth("UnitedStates", dates)
endOfMonth("UnitedStates/Settlement", dates)      ## same as previous
endOfMonth("UnitedStates/NYSE", dates)            ## stocks
endOfMonth("UnitedStates/GovernmentBond", dates)  ## bonds
endOfMonth("UnitedStates/NERC", dates)            ## energy
```

---

Enum                          *Documentation for parameters*

---

## Description

Reference for parameters when constructing a bond

## Arguments

`DayCounter`      an int value

| | |
|---|---|
| 0 | Actual360 |
| 1 | Actual360FixEd |
| 2 | ActualActual |
| 3 | ActualBusiness252 |
| 4 | OneDayCounter |
| 5 | SimpleDayCounter |
| anything else | Thirty360 |

`businessDayConvention`
                 an int value

| | |
|---|---|
| 0 | Following |
| 1 | ModifiedFollowing |
| 2 | Preceding |
| 3 | ModifiedPreceding |
| anything else | UNadjusted |

`compounding`     an int value

| | |
|---|---|
| 0 | Simple |
| 1 | Compounded |
| 2 | Continuous |
| 3 | SimpleThenCompounded |

`period or frequency`

an int value

| | |
|---|---|
| `-1` | NoFrequency |
| `0` | Once |
| `1` | Annual |
| `2` | Semiannual |
| `3` | EveryFourthMonth |
| `4` | Quarterly |
| `6` | BiMonthtly |
| `12` | Monthly |
| `13` | EveryFourthWeek |
| `26` | BiWeekly |
| `52` | Weekly |
| `365` | Daily |
| `anything else` | OtherFrequency |

`date generation`

an int value to specify date generation rule

| | |
|---|---|
| `0` | Backward |
| `1` | Forward |
| `2` | Zero |
| `3` | ThirdWednesday |
| `4` | Twentieth |
| `anything else` | TwentiethIMM |

## Details

http://quantlib.org/reference/class_quant_lib_1_1_day_counter.html
http://quantlib.org/reference/group__datetime.html

## Value

None

## Author(s)

Khanh Nguyen <knguyen@cs.umb.edu>

## References

http://quantlib.org for details on `QuantLib`.

---

EuropeanOption              *European Option evaluation using Closed-Form solution*

---

### Description

The `EuropeanOption` function evaluations an European-style option on a common stock using the Black-Scholes-Merton solution. The option value, the common first derivatives ("Greeks") as well as the calling parameters are returned.

### Usage

```
## Default S3 method:
EuropeanOption(type, underlying, strike,
dividendYield, riskFreeRate, maturity, volatility)

## S3 method for class 'Option':
plot
## S3 method for class 'Option':
print
## S3 method for class 'Option':
summary
```

### Arguments

| | |
|---|---|
| `type` | A string with one of the values `call` or `put` |
| `underlying` | Current price of the underlying stock |
| `strike` | Strike price of the option |
| `dividendYield` | |
| | Continuous dividend yield (as a fraction) of the stock |
| `riskFreeRate` | Risk-free rate |
| `maturity` | Time to maturity (in fractional years) |
| `volatility` | Volatility of the underlying stock |

### Details

The well-known closed-form solution derived by Black, Scholes and Merton is used for valuation. Implied volatilities are calculated numerically.

Please see any decent Finance textbook for background reading, and the `QuantLib` documentation for details on the `QuantLib` implementation.

### Value

The `EuropeanOption` function returns an object of class `EuropeanOption` (which inherits from class `Option`). It contains a list with the following components:

| | |
|---|---|
| `value` | Value of option |

| | |
|---|---|
| delta | Sensitivity of the option value for a change in the underlying |
| gamma | Sensitivity of the option delta for a change in the underlying |
| vega | Sensitivity of the option value for a change in the underlying's volatility |
| theta | Sensitivity of the option value for a change in t, the remaining time to maturity |
| rho | Sensitivity of the option value for a change in the risk-free interest rate |
| dividendRho | Sensitivity of the option value for a change in the dividend yield |
| parameters | List with parameters with which object was created |

## Note

The interface might change in future release as `QuantLib` stabilises its own API.

## Author(s)

Dirk Eddelbuettel <edd@debian.org> for the R interface; the QuantLib Group for `QuantLib`

## References

<http://quantlib.org> for details on `QuantLib`.

## See Also

`EuropeanOptionImpliedVolatility`, `EuropeanOptionArrays`, `AmericanOption`, `BinaryOption`

## Examples

```
# simple call with unnamed parameters
EuropeanOption("call", 100, 100, 0.01, 0.03, 0.5, 0.4)
# simple call with some explicit parameters, and slightly increased vol:
EuropeanOption(type="call", underlying=100, strike=100, dividendYield=0.01,
riskFreeRate=0.03, maturity=0.5, volatility=0.5)
```

---

EuropeanOptionArrays

*European Option evaluation using Closed-Form solution*

---

## Description

The `EuropeanOptionArrays` function allows any of the numerical input parameters to be a list, and a list of arrays is returned. Each of the returned arrays has as many dimension as there were lists among the input parameters, and each multi-dimensional array element corresponds to an evaluation under the given set of parameters.

## Usage

```
EuropeanOptionArrays(type, underlying, strike, dividendYield, riskFreeRate, maturit
```

**Arguments**

| | |
|---|---|
| `type` | A string with one of the values `call` or `put` |
| `underlying` | (Scalar or list) current price(s) of the underlying stock |
| `strike` | (Scalar or list) strike price(s) of the option |
| `dividendYield` | |
| | (Scalar or list) continuous dividend yield(s) (as a fraction) of the stock |
| `riskFreeRate` | (Scalar or list) risk-free rate(s) |
| `maturity` | (Scalar or list) time(s) to maturity (in fractional years) |
| `volatility` | (Scalar or list) volatilit(y|ies) of the underlying stock |

**Details**

The well-known closed-form solution derived by Black, Scholes and Merton is used for valuation.

Please see any decent Finance textbook for background reading, and the `QuantLib` documentation for details on the `QuantLib` implementation.

**Value**

The `EuropeanOptionArrays` function allows each of the numerical input parameters to be a list (or vector, or sequence). A list of multi-dimensional arrays is returned. Each array point corresponds to an evaluation under the given set of parameters.

For these functions, the following components are returned:

| | |
|---|---|
| `value` | (Scalar or array) value of option |
| `delta` | (Scalar or array) change in value for a change in the underlying |
| `gamma` | (Scalar or array) change in value for a change in delta |
| `vega` | (Scalar or array) change in value for a change in the underlying's volatility |
| `theta` | (Scalar or array) change in value for a change in delta |
| `rho` | (Scalar or array) change in value for a change in time to maturity |
| `dividendRho` | (Scalar or array) change in value for a change in delta |
| `parameters` | List with parameters with which object was created |

**Note**

The interface might change in future release as `QuantLib` stabilises its own API.

**Author(s)**

Dirk Eddelbuettel <edd@debian.org> for the R interface; the QuantLib Group for `QuantLib`

**References**

http://quantlib.org for details on `QuantLib`.

## See Also

AmericanOption,BinaryOption

## Examples

```
# define two vectos for the underlying and the volatility
und.seq <- seq(10,180,by=2)
vol.seq <- seq(0.1,0.9,by=0.1)
# evaluate them along with three scalar parameters
EOarr <- EuropeanOptionArrays("call", underlying=und.seq,
                              strike=100, dividendYield=0.01,
                              riskFreeRate=0.03,
                              maturity=1, volatility=vol.seq)
# and look at four of the result arrays: value, delta, gamma, vega
old.par <- par(no.readonly = TRUE)
par(mfrow=c(2,2),oma=c(5,0,0,0),mar=c(2,2,2,1))
plot(EOarr$parameter$underlying, EOarr$value[,1], type='n',
     main="option value", xlab="", ylab="")
topocol <- topo.colors(length(vol.seq))
for (i in 1:length(vol.seq))
  lines(EOarr$parameter$underlying, EOarr$value[,i], col=topocol[i])
plot(EOarr$parameter$underlying, EOarr$delta[,1],type='n',
     main="option delta", xlab="", ylab="")
for (i in 1:length(vol.seq))
  lines(EOarr$parameter$underlying, EOarr$delta[,i], col=topocol[i])
plot(EOarr$parameter$underlying, EOarr$gamma[,1],type='n',
     main="option gamma", xlab="", ylab="")
for (i in 1:length(vol.seq))
  lines(EOarr$parameter$underlying, EOarr$gamma[,i], col=topocol[i])
plot(EOarr$parameter$underlying, EOarr$vega[,1],type='n',
     main="option vega", xlab="", ylab="")
for (i in 1:length(vol.seq))
  lines(EOarr$parameter$underlying, EOarr$vega[,i], col=topocol[i])
mtext(text=paste("Strike is 100, maturity 1 year, riskless rate 0.03",
        "\nUnderlying price from", und.seq[1],"to", und.seq[length(und.seq)],
        "\nVolatility  from",vol.seq[1], "to",vol.seq[length(vol.seq)]),
      side=1,font=1,outer=TRUE,line=3)
par(old.par)
```

---

EuropeanOptionImpliedVolatility
*Implied Volatility calculation for European Option*

---

## Description

The `EuropeanOptionImpliedVolatility` function solves for the (unobservable) implied
volatility, given an option price as well as the other required parameters to value an option.

**Usage**

```
## Default S3 method:
EuropeanOptionImpliedVolatility(type, value,
underlying, strike, dividendYield, riskFreeRate, maturity, volatility)

## S3 method for class 'ImpliedVolatility':
print
## S3 method for class 'ImpliedVolatility':
summary
```

**Arguments**

| | |
|---|---|
| type | A string with one of the values call or put |
| value | Value of the option (used only for ImpliedVolatility calculation) |
| underlying | Current price of the underlying stock |
| strike | Strike price of the option |
| dividendYield | |
| | Continuous dividend yield (as a fraction) of the stock |
| riskFreeRate | Risk-free rate |
| maturity | Time to maturity (in fractional years) |
| volatility | Initial guess for the volatility of the underlying stock |

**Details**

The well-known closed-form solution derived by Black, Scholes and Merton is used for valuation. Implied volatilities are then calculated numerically.

Please see any decent Finance textbook for background reading, and the QuantLib documentation for details on the QuantLib implementation.

**Value**

The EuropeanOptionImpliedVolatility function returns an object of class ImpliedVolatility. It contains a list with the following elements:

| | |
|---|---|
| impliedVol | The volatility implied by the given market prices |
| parameters | List with the option parameters used |

**Note**

The interface might change in future release as QuantLib stabilises its own API.

**Author(s)**

Dirk Eddelbuettel <edd@debian.org> for the R interface; the QuantLib Group for QuantLib

## References

<http://quantlib.org> for details on `QuantLib`.

## See Also

`EuropeanOption`,`AmericanOption`,`BinaryOption`

## Examples

```
EuropeanOptionImpliedVolatility(type="call", value=11.10, underlying=100,
strike=100, dividendYield=0.01, riskFreeRate=0.03,
maturity=0.5, volatility=0.4)
```

---

| FittedBondCurve | *Returns the discount curve (with zero rates and forwards) given set of bonds* |
|---|---|

---

## Description

`FittedBondCurve` fits a term structure to a set of bonds using three different fitting methodologies. For more detail, see QuantLib/Example/FittedBondCurve.

## Usage

```
FittedBondCurve(curveparams, lengths, coupons, dateparams)
```

## Arguments

`curveparams`  curve parameters

| | | |
|---|---|---|
| `method` | a string, fitting methods: "ExponentialSplinesFitting", "SimplePolynomialFitting", "NelsonSiegelFitting" |
| `origDate` | a Date, starting date of the curve |

`lengths`  an numeric vector, length of the bonds in year

`coupons`  a numeric vector, coupon rate of the bonds

`dateparams`  (Optional) a named list, QuantLib's date parameters of the bond.

| | |
|---|---|
| `settlementDays` | (Optional) a double, settlement days. Default value is 1. |
| `dayCounter` | (Optional) a number or string, day counter convention. See Enum. Default value is 'Thirty360' |
| `period` | (Optional) a number or string, interest compounding interval. See Enum. Default value is 'Semiannual'. |

|                         |                                                                    |
|-------------------------|--------------------------------------------------------------------|
| businessDayConvention   | (Optional) a number or string, business day convention. See Enum. Default value is 'Following'. |

See example below.

## Details

Please see any decent Finance textbook for background reading, and the `QuantLib` documentation for details on the `QuantLib` implementation.

## Value

`table`, a three columns "date - zeroRate - discount" data frame

## Author(s)

Khanh Nguyen <knguyen@cs.umb.edu> for the inplementation; Dirk Eddelbuettel <edd@debian.org> for the R interface; the QuantLib Group for `QuantLib`

## References

http://quantlib.org/reference/class_quant_lib_1_1_fitted_bond_discount_curve.html

## Examples

```
lengths <- c(2,4,6,8,10,12,14,16,18,20,22,24,26,28,30)
coupons <- c( 0.0200, 0.0225, 0.0250, 0.0275, 0.0300,
              0.0325, 0.0350, 0.0375, 0.0400, 0.0425,
              0.0450, 0.0475, 0.0500, 0.0525, 0.0550 )
dateparams <- list(settlementDays=0, period="Annual",
                   dayCounter="SimpleDayCounter",
                   businessDayConvention ="Unadjusted")
curveparams <- list(method="ExponentialSplinesFitting",
                    origDate = Sys.Date())
curve <- FittedBondCurve(curveparams, lengths, coupons, dateparams)
library(zoo)
z <- zoo(curve$table$zeroRates, order.by=curve$table$date)
plot(z)
```

---

```
FixedRateBond        Fixed-Rate bond pricing
```

---

### Description

The `FixedRateBond` function evaluates a fixed rate bond using discount curve. More specificly, the calculation is done by DiscountingBondEngine from QuantLib. The NPV, clean price, dirty price, accrued interest, yield and cash flows of the bond is returned. For more detail, see the source codes in quantlib's test-suite. test-suite/bond.cpp

### Usage

```
## Default S3 method:
FixedRateBond(bond, rates, discountCurve, dateparams )
## S3 method for class 'Bond':
plot
## S3 method for class 'Bond':
print
## S3 method for class 'Bond':
summary
```

### Arguments

bond            bond parameters, a named list whose elements are:

| | |
|---|---|
| issueDate | a Date, the bond's issue date |
| maturityDate | a Date, the bond's maturity date |
| faceAmount | (Optional) a double, face amount of the bond. Default value is 100. |
| redemption | (Optional) a double, percentage of the initial face amount that will be returned at maturity date. Default value is 100. |
| effectiveDate | (Optinal) a Date, the bond's effective date. Default value is issueDate |

rates           a numeric vector, bond's coupon rates

discountCurve

        Can be one of the following:

| | |
|---|---|
| a DiscountCurve | a object of DiscountCurve class For more detail, see example or the discountCurve function |
| A 2 items list | specifies a flat curve in two values "todayDate" and "rate" |
| A 3 items list | specifies three values to construct a DiscountCurve object, "params" , "tsQuotes", "times". |

For more detail, see example or
the discountCurve function

| dateparams | (Optional) a named list, QuantLib's date parameters of the bond. |
| --- | --- |
| settlementDays | (Optional) a double, settlement days. Default value is 1. |
| calendar | (Optional) a string, either 'us' or 'uk' corresponding to US Goverment Bond calendar and UK Exchange calendar. Default value is 'us'. |
| dayCounter | (Optional) a number or string, day counter convention. See [Enum]. Default value is 'Thirty360' |
| period | (Optional) a number or string, interest compounding interval. See [Enum]. Default value is 'Semiannual'. |
| businessDayConvention | (Optional) a number or string, business day convention. See [Enum]. Default value is 'Following'. |
| terminationDateConvention | (Optional) a number or string, termination day convention. See [Enum]. Default value is 'Following'. |
| endOfMonth | (Optional) a numeric with value 1 or 0. End of Month rule. Default value is 0. |
| dateGeneration | (Optional) a numeric, date generation method. See [Enum]. Default value is 'Backward' |

See example below.

## Details

A discount curve is built to calculate the bond value.

Please see any decent Finance textbook for background reading, and the QuantLib documentation for details on the QuantLib implementation.

## Value

The FixedRateBond function returns an object of class FixedRateBond (which inherits from class Bond). It contains a list with the following components:

| | |
| --- | --- |
| NPV | net present value of the bond |
| cleanPrice | clean price of the bond |
| dirtyPrice | dirty price of the bond |
| accruedAmount | |
| | accrued amount of the bond |

yield         yield of the bond

cashFlows     cash flows of the bond

### Note

The interface might change in future release as QuantLib stabilises its own API.

### Author(s)

Khanh Nguyen <knguyen@cs.umb.edu> for the inplementation; Dirk Eddelbuettel <edd@debian.org> for the R interface; the QuantLib Group for QuantLib

### References

http://quantlib.org for details on QuantLib.

### Examples

```
bond <- list(faceAmount=100, issueDate=as.Date("2004-11-30"),
             maturityDate=as.Date("2008-11-30"), redemption=100,
             effectiveDate=as.Date("2004-11-30"))
dateparams <- list(settlementDays=1, calendar="us", dayCounter = 1, period=3,
                   businessDayConvention = 4, terminationDateConvention=4,
                   dateGeneration=1, endOfMonth=1)
curve <- list(todayDate=as.Date("2004-11-04"), riskFreeRate=0.03)
rates <- c(0.02875)

FixedRateBond(bond, rates, curve, dateparams)

params <- list(tradeDate=as.Date('2002-2-15'),
               settleDate=as.Date('2002-2-19'),
               dt=.25,
               interpWhat="discount",
               interpHow="loglinear")

tsQuotes <- list(d1w  =0.0382,
                 d1m  =0.0372,
                 fut1=96.2875,
                 fut2=96.7875,
                 fut3=96.9875,
                 fut4=96.6875,
                 fut5=96.4875,
                 fut6=96.3875,
                 fut7=96.2875,
                 fut8=96.0875,
                 s3y  =0.0398,
                 s5y  =0.0443,
                 s10y =0.05165,
                 s15y =0.055175)

times <- seq(0,10,.1)
```

```
curve <- list(params, tsQuotes, times)
FixedRateBond(bond, rates, curve, dateparams)

curve <- DiscountCurve(params, tsQuotes, times)
dateparams <- list(settlementDays=1, calendar="us", dayCounter = "Thirty360",
                   period="Annual", businessDayConvention = "Preceding",
                   terminationDateConvention="Preceding",
                   dateGeneration="Forward", endOfMonth=1)
FixedRateBond(bond, rates, curve, dateparams)


#example with default dateparams
FixedRateBond(bond, rates, curve)

##exampe with defaul bond parameter and dateparams
bond <- list(issueDate=as.Date("2004-11-30"),
maturityDate=as.Date("2008-11-30"))
dateparams <- list(calendar="us", dayCounter = "ActualActual",
                   period="Annual")
FixedRateBond(bond, rates, curve, dateparams)
```

---

FixedRateBondPriceByYield

*Zero Coupon Bond Yield evaluation*

---

### Description

The `FixedRateBondPriceByYield` function calculates the theoretical price of a fixed rate bond from its yield

### Usage

```
## Default S3 method:
FixedRateBondPriceByYield( settlementDays=1, yield, faceAmount,
                                effectiveDate, maturityDate,
                                period, calendar="us",
                                rates, dayCounter=2,
                                businessDayConvention=0, compound = 0, redemption=
                                issueDate)

## S3 method for class 'Bond':
plot
## S3 method for class 'Bond':
print
## S3 method for class 'Bond':
summary
```

## Arguments

settlementDays

an integer, 1 for T+1, 2 for T+2, etc...

yield          yield of the bond

effectiveDate

bond's effective date

maturityDate   bond's maturity date

period         frequency of events,0=NoFrequency, 1=Once, 2=Annual, 3=Semiannual, 4=EveryFourthMonth, 5=Quarterly, 6=Bimonthly ,7=Monthly ,8=EveryFourthWeely,9=Biweekly, 10=Weekly, 11=Daily. For more information, see QuantLib's Frequency class

calendar       Business Calendar. Either us or uk

faceAmount     face amount of the bond

rates          vector of rates

businessDayConvention

convention used to adjust a date in case it is not a valid business day. See quantlib for more detail. 0 = Following, 1 = ModifiedFollowing, 2 = Preceding, 3 = ModifiedPreceding, other = Unadjusted

dayCounter     day count convention. 0 = Actual360(), 1 = Actual365Fixed(), 2 = ActualActual(), 3 = Business252(), 4 = OneDayCounter(), 5 = SimpleDayCounter(), all other = Thirty360(). For more information, see QuantLib's DayCounter class

compound       compounding type. 0=Simple, 1=Compounded, 2=Continuous, all other=SimpleThenCompounded. See QuantLib's Compound class

redemption     redemption when the bond expires

issueDate      date the bond is issued

## Value

The FixedRateBondPriceByYield function returns an object of class FixedRateBondPriceByYield (which inherits from class Bond). It contains a list with the following components:

yield          yield of the bond

## Note

The interface might change in future release as QuantLib stabilises its own API.

## Author(s)

Khanh Nguyen <knguyen@cs.umb.edu>

## References

http://quantlib.org for details on QuantLib. http://www.mathworks.com/access/helpdesk/help/toolbox/finfixed/FixedRateBondPriceByYield.html for more details about this function

## Examples

```
FixedRateBondPriceByYield(,0.0307, 100000, as.Date("2004-11-30"), as.Date("2008-11-30"), 3,
```

---

FixedRateBondYield *Fixed Rate Bond Yield Yield evaluation*

---

## Description

The `FixedRateBondYield` function calculates the theoretical yield of a fixed rate bond from
its price

## Usage

```
## Default S3 method:
FixedRateBondYield( settlementDays=1, price, faceAmount,
                                effectiveDate, maturityDate,
                                period, calendar="us",
                                rates, dayCounter=2,
                                businessDayConvention=0,
                                compound = 0, redemption=100,
                                issueDate)

## S3 method for class 'Bond':
plot
## S3 method for class 'Bond':
print
## S3 method for class 'Bond':
summary
```

## Arguments

settlementDays
>               an integer, 1 for T+1, 2 for T+2, etc...

price           price of the bond

effectiveDate
>               bond's effective date

maturityDate   bond's maturity date

period         frequency of events,0=NoFrequency, 1=Once, 2=Annual, 3=Semiannual, 4=EveryFourthMonth, 5=Quarterly, 6=Bimonthly ,7=Monthly ,8=EveryFourthWeely,9=Biweekly, 10=Weekly, 11=Daily. For more information, see QuantLib's Frequency class

calendar       Business Calendar. Either us or uk

faceAmount     face amount of the bond

rates          vector of rates

| | |
|---|---|
| businessDayConvention | |
| | convention used to adjust a date in case it is not a valid business day. See quantlib for more detail. 0 = Following, 1 = ModifiedFollowing, 2 = Preceding, 3 = ModifiedPreceding, other = Unadjusted |
| dayCounter | day count convention. 0 = Actual360(), 1 = Actual365Fixed(), 2 = ActualActual(), 3 = Business252(), 4 = OneDayCounter(), 5 = SimpleDayCounter(), all other = Thirty360(). For more information, see QuantLib's DayCounter class |
| compound | compounding type. 0=Simple, 1=Compounded, 2=Continuous, all other=SimpleThenCompounded. See QuantLib's Compound class |
| redemption | redemption when the bond expires |
| issueDate | date the bond is issued |

## Value

The `FixedRateBondYield` function returns an object of class `FixedRateBondYield` (which inherits from class `Bond`). It contains a list with the following components:

| | |
|---|---|
| yield | yield of the bond |

## Note

The interface might change in future release as `QuantLib` stabilises its own API.

## Author(s)

Khanh Nguyen <knguyen@cs.umb.edu>

## References

http://quantlib.org for details on `QuantLib`. http://www.mathworks.com/access/helpdesk/help/toolbox/finfixed/FixedRateBondYield.html for more details about this function

## Examples

```
FixedRateBondYield(,90, 100000, as.Date("2004-11-30"), as.Date("2008-11-30"), 3, , c(0.02875
```

---

FloatingRateBond    *Floating rate bond pricing*

---

## Description

The `FloatingRateBond` function evaluates a floating rate bond using discount curve. More specificly, the calculation is done by DiscountingBondEngine from QuantLib. The NPV, clean price, dirty price, accrued interest, yield and cash flows of the bond is returned. For more detail, see the source codes in quantlib's test-suite. test-suite/bond.cpp

**Usage**

```
## Default S3 method:
FloatingRateBond(bond, gearings, spreads,
                                  caps, floors, index,
                                  curve, dateparams )
## S3 method for class 'Bond':
plot
## S3 method for class 'Bond':
print
## S3 method for class 'Bond':
summary
```

**Arguments**

bond            bond parameters, a named list whose elements are:

| | |
|---|---|
| issueDate | a Date, the bond's issue date |
| maturityDate | a Date, the bond's maturity date |
| faceAmount | (Optional) a double, face amount of the bond. Default value is 100. |
| redemption | (Optional) a double, percentage of the initial face amount that will be returned at maturity date. Default value is 100. |
| effectiveDate | (Optinal) a Date, the bond's effective date. Default value is issueDate |

gearings        (Optional) a numeric vector, bond's gearings. See quantlib's doc on FloatingRateBond for more detail. Default value is an empty vector c().

spreads         (Optional) a numeric vector, bond's spreads. See quantlib's doc on FloatingRateBond for more detail.Default value is an empty vector c()

caps            (Optional) a numeric vector, bond's caps. See quantlib's doc on FloatingRateBond for more detail. Default value is an empty vector c()

floors          (Optional) a numeric vector, bond's floors. See quantlib's doc on FloatingRateBond for more detail. Default value is an empty vector c()

curve           Can be one of the following:

| | |
|---|---|
| a DiscountCurve | a object of DiscountCurve class For more detail, see example or the discountCurve function |
| A 2 items list | specifies a flat curve in two values "todayDate" and "rate" |
| A 3 items list | specifies three values to construct a DiscountCurve object, "params" , "tsQuotes", "times". For more detail, see example or the discountCurve function |

| index | a named list whose elements are parameters of an IborIndex term structure. |
|---|---|

| | | |
|---|---|---|
| | type | a string, currently support only "USDLibor" |
| | length | an integer, length of the index |
| | inTermOf | a string, period unit, currently support only 'Month' |
| | term | a DiscountCurve object, the term structure of the index |

| dateparams | (Optional) a named list, QuantLib's date parameters of the bond. |
|---|---|

| | |
|---|---|
| settlementDays | (Optional) a double, settlement days. Default value is 1. |
| calendar | (Optional) a string, either 'us' or 'uk' corresponding to US Goverment Bond calendar and UK Exchange calendar. Default value is 'us'. |
| dayCounter | (Optional) a number or string, day counter convention. See [Enum]. Default value is 'Thirty360' |
| period | (Optional) a number or string, interest compounding interval. See [Enum]. Default value is 'Semiannual'. |
| businessDayConvention | (Optional) a number or string, business day convention. See [Enum]. Default value is 'Following'. |
| terminationDateConvention | (Optional) a number or string, termination day convention. See [Enum]. Default value is 'Following'. |
| endOfMonth | (Optional) a numeric with value 1 or 0. End of Month rule. Default value is 0. |
| dateGeneration | (Optional) a numeric, date generation method. See [Enum]. Default value is 'Backward' |

See example below.

### Details

A discount curve is built to calculate the bond value.

Please see any decent Finance textbook for background reading, and the QuantLib documentation for details on the QuantLib implementation.

### Value

The FloatingRateBond function returns an object of class FloatingRateBond (which inherits from class Bond). It contains a list with the following components:

| NPV | net present value of the bond |
|---|---|
| cleanPrice | clean price of the bond |

dirtyPrice      dirty price of the bond

accruedAmount

                accrued amount of the bond

yield           yield of the bond

cashFlows       cash flows of the bond

## Note

The interface might change in future release as `QuantLib` stabilises its own API.

## Author(s)

Khanh Nguyen <knguyen@cs.umbno.edu> for the inplementation; Dirk Eddelbuettel <edd@debian.org>
for the R interface; the QuantLib Group for `QuantLib`

## References

<http://quantlib.org> for details on `QuantLib`.

## Examples

```
bond <- list(faceAmount=100, issueDate=as.Date("2004-11-30"),
             maturityDate=as.Date("2008-11-30"), redemption=100,
             effectiveDate=as.Date("2004-11-30"))

dateparams <- list(settlementDays=1, calendar="us", dayCounter = 1, period=3,
                   businessDayConvention = 1, terminationDateConvention=1,
                   dateGeneration=0, endOfMonth=0, fixingDays = 1)

gearings <- c()

spreads <- c()

caps <- c()

floors <- c()


length2 <- list(todayDate=as.Date("2004-11-22"), riskFreeRate=0.025)

params <- list(tradeDate=as.Date('2002-2-15'),
               settleDate=as.Date('2002-2-19'),
               dt=.25,
               interpWhat="discount",
               interpHow="loglinear")

tsQuotes <- list(d1w  =0.0382,
                 d1m  =0.0372,
                  fut1=96.2875,
                  fut2=96.7875,
```

```
                      fut3=96.9875,
                      fut4=96.6875,
                      fut5=96.4875,
                      fut6=96.3875,
                      fut7=96.2875,
                      fut8=96.0875,
                      s3y  =0.0398,
                      s5y  =0.0443,
                      s10y =0.05165,
                      s15y =0.055175)

times <- seq(0,10,.1)

length3 <- list(params, tsQuotes, times)


# both curves are flat

curve <- length2
termstructure <- length2
iborindex <- list(type="USDLibor", length=6,
                  inTermOf="Month", term=termstructure)
FloatingRateBond(bond, gearings, spreads, caps, floors,
                 iborindex, curve, dateparams)


# one flat, another one is constructe

curve <- length2
termstructure <- length3
iborindex <- list(type="USDLibor", length=6,
                  inTermOf="Month", term = termstructure)
FloatingRateBond(bond, gearings, spreads, caps, floors,
                 iborindex, curve, dateparams)


curve <- length3
termstructure <- length2
iborindex <- list(type="USDLibor", length=6,
                  inTermOf="Month", term = termstructure)
FloatingRateBond(bond, gearings, spreads, caps, floors,
                 iborindex, curve, dateparams)


# both curves are constructed

curve <- length3
termstructure <- length3
iborindex <- list(type="USDLibor", length=6,
                  inTermOf="Month", term = termstructure)
FloatingRateBond(bond, gearings, spreads, caps, floors,
                 iborindex, curve, dateparams)
```

```
curve2 <- DiscountCurve(params, tsQuotes, times)
index2 <- DiscountCurve(params, tsQuotes, times)
ibor <- list(type="USDLibor", length=6,
             inTermOf="Month", term = index2)
dateparams <- list(settlementDays=1, calendar="us", dayCounter = "Actual360",
                   period="Semiannual",
                   businessDayConvention = "Following",
                   terminationDateConvention= "Following",
                   dateGeneration= "Forward",
                   endOfMonth=0, fixingDays = 1)
FloatingRateBond(bond, gearings, spreads, caps, floors,
                 ibor, curve2, dateparams)

#example using default values
FloatingRateBond(bond=bond, index=ibor, curve=curve2)
```

---

holidayList                     *Calendar functions from QuantLib*

---

### Description

The `holidayList` function evaluates two given dates in the context of the given calendar, and returns a vector that gives the list of holiday between.

### Usage

```
holidayList(calendar="TARGET", from=Sys.Date(),
to = Sys.Date() + 5, includeWeekends = 0)
```

### Arguments

| | |
|---|---|
| calendar | A string identifying one of the supported QuantLib calendars, see Details for more |
| from | A vector (or scalar) of `Date` types. |
| to | A vector (or scalar) of `Date` types. |
| includeWeekends | |
| | boolean that indicates whether the calculation should include the weekends. Default = false |

### Details

The calendars are coming from QuantLib, and the QuantLib documentation should be consulted for details.

Currently, the following strings are recognised: TARGET (a default calendar), Canada and Canada/Settlement, Canada/TSX, Germany and Germany/FrankfurtStockExchange, Germany/Settlement, Germany/Xetra,

Germany/Eurex, Italy and Italy/Settlement, Italy/Exchange, Japan, UnitedKingdom and United-Kingdom/Settlement, UnitedKingdom/Exchange, UnitedKingdom/Metals, UnitedStates and UnitedStates/Settlement, UnitedStates/NYSE, UnitedStates/GovernmentBond, UnitedStates/NERC.

(In case of multiples entries per country, the country default is listed right after the country itself. Using the shorter form is equivalent.)

### Value

An vector of dates.

### Note

The interface might change in future release as `QuantLib` stabilises its own API.

### Author(s)

Dirk Eddelbuettel <edd@debian.org> for the R interface; Khanh Nguyen <nguyen.h.khanh@gmail.com> for the implementation; the QuantLib Group for `QuantLib`

### References

<http://quantlib.org> for details on `QuantLib`.

### Examples

```
  from <- as.Date("2009-04-07")
to<-as.Date("2009-04-14")
holidayList("UnitedStates", from, to)
to <- as.Date("2009-10-7")
holidayList("UnitedStates", from, to)
```

---

ImpliedVolatility    *Base class for option-price implied volatility evalution*

---

### Description

This class forms the basis from which the more specific classes are derived.

### Usage

```
## S3 method for class 'ImpliedVolatility':
print
## S3 method for class 'ImpliedVolatility':
summary
```

### Arguments

Any option-price implied volatility object derived from this base class

## Details

Please see any decent Finance textbook for background reading, and the `QuantLib` documentation for details on the `QuantLib` implementation.

## Value

None, but side effects of displaying content.

## Note

The interface might change in future release as `QuantLib` stabilises its own API.

## Author(s)

Dirk Eddelbuettel <edd@debian.org> for the R interface; the QuantLib Group for `QuantLib`

## References

<http://quantlib.org> for details on `QuantLib`.

## See Also

`AmericanOptionImpliedVolatility`, `EuropeanOptionImpliedVolatility`, `AmericanOption`, `Europ`
`BinaryOption`

## Examples

```
impVol<-EuropeanOptionImpliedVolatility("call", value=11.10, strike=100, volatility=0.4, 100
print(impVol)
summary(impVol)
```

---

isEndOfMonth          *Calendar functions from QuantLib*

---

## Description

The `isEndOfMonth` function evaluates the given dates in the context of the given calendar, and returns a vector of booleans indicating end of month status.

## Usage

```
isEndOfMonth(calendar="TARGET", dates=Sys.Date())
```

## Arguments

| | |
|---|---|
| calendar | A string identifying one of the supported QuantLib calendars, see Details for more |
| dates | A vector (or scalar) of `Date` types. |

## Details

The calendars are coming from QuantLib, and the QuantLib documentation should be consulted for details.

Currently, the following strings are recognised: TARGET (a default calendar), Canada and Canada/Settlement, Canada/TSX, Germany and Germany/FrankfurtStockExchange, Germany/Settlement, Germany/Xetra, Germany/Eurex, Italy and Italy/Settlement, Italy/Exchange, Japan, UnitedKingdom and United-Kingdom/Settlement, UnitedKingdom/Exchange, UnitedKingdom/Metals, UnitedStates and Unit-edStates/Settlement, UnitedStates/NYSE, UnitedStates/GovernmentBond, UnitedStates/NERC.

(In case of multiples entries per country, the country default is listed right after the country itself. Using the shorter form is equivalent.)

## Value

An named vector of booleans each of which is true if the corresponding date is an end of month in the given calendar. The element names are the dates (formatted as text in yyyy-mm-dd format).

## Note

The interface might change in future release as `QuantLib` stabilises its own API.

## Author(s)

Dirk Eddelbuettel <edd@debian.org> for the R interface; Khanh Nguyen <nguyen.h.khanh@gmail.com> for the implementation; the QuantLib Group for `QuantLib`

## References

http://quantlib.org for details on `QuantLib`.

## Examples

```
dates <- seq(from=as.Date("2009-04-07"), to=as.Date("2009-04-14"), by=1)
isEndOfMonth("UnitedStates", dates)
isEndOfMonth("UnitedStates/Settlement", dates)      ## same as previous
isEndOfMonth("UnitedStates/NYSE", dates)            ## stocks
isEndOfMonth("UnitedStates/GovernmentBond", dates)  ## bonds
isEndOfMonth("UnitedStates/NERC", dates)            ## energy
```

---

isHoliday                 *Calendar functions from QuantLib*

---

## Description

The `isHoliday` function evaluates the given dates in the context of the given calendar, and returns a vector of booleans indicating holiday day status.

## Usage

```
isHoliday(calendar="TARGET", dates=Sys.Date())
```

## Arguments

| | |
|---|---|
| calendar | A string identifying one of the supported QuantLib calendars, see Details for more |
| dates | A vector (or scalar) of Date types. |

## Details

The calendars are coming from QuantLib, and the QuantLib documentation should be consulted for details.

Currently, the following strings are recognised: TARGET (a default calendar), Canada and Canada/Settlement, Canada/TSX, Germany and Germany/FrankfurtStockExchange, Germany/Settlement, Germany/Xetra, Germany/Eurex, Italy and Italy/Settlement, Italy/Exchange, Japan, UnitedKingdom and UnitedKingdom/Settlement, UnitedKingdom/Exchange, UnitedKingdom/Metals, UnitedStates and UnitedStates/Settlement, UnitedStates/NYSE, UnitedStates/GovernmentBond, UnitedStates/NERC.

(In case of multiples entries per country, the country default is listed right after the country itself. Using the shorter form is equivalent.)

## Value

An named vector of booleans each of which is true if the corresponding date is a holiday day in the given calendar. The element names are the dates (formatted as text in yyyy-mm-dd format).

## Note

The interface might change in future release as QuantLib stabilises its own API.

## Author(s)

Dirk Eddelbuettel <edd@debian.org> for the R interface; Khanh Nguyen <nguyen.h.khanh@gmail.com> for the implementation; the QuantLib Group for QuantLib

## References

http://quantlib.org for details on QuantLib.

## Examples

```
dates <- seq(from=as.Date("2009-04-07"), to=as.Date("2009-04-14"), by=1)
isHoliday("UnitedStates", dates)
isHoliday("UnitedStates/Settlement", dates)      ## same as previous
isHoliday("UnitedStates/NYSE", dates)            ## stocks
isHoliday("UnitedStates/GovernmentBond", dates)  ## bonds
isHoliday("UnitedStates/NERC", dates)            ## energy
```

---

isWeekend *Calendar functions from QuantLib*

---

### Description

The `isWeekend` function evaluates the given dates in the context of the given calendar, and returns a vector of booleans indicating weekend status.

### Usage

```
isWeekend(calendar="TARGET", dates=Sys.Date())
```

### Arguments

calendar   A string identifying one of the supported QuantLib calendars, see Details for more

dates      A vector (or scalar) of Date types.

### Details

The calendars are coming from QuantLib, and the QuantLib documentation should be consulted for details.

Currently, the following strings are recognised: TARGET (a default calendar), Canada and Canada/Settlement, Canada/TSX, Germany and Germany/FrankfurtStockExchange, Germany/Settlement, Germany/Xetra, Germany/Eurex, Italy and Italy/Settlement, Italy/Exchange, Japan, UnitedKingdom and UnitedKingdom/Settlement, UnitedKingdom/Exchange, UnitedKingdom/Metals, UnitedStates and UnitedStates/Settlement, UnitedStates/NYSE, UnitedStates/GovernmentBond, UnitedStates/NERC.

(In case of multiples entries per country, the country default is listed right after the country itself. Using the shorter form is equivalent.)

### Value

An named vector of booleans each of which is true if the corresponding date is a weekend in the given calendar. The element names are the dates (formatted as text in yyyy-mm-dd format).

### Note

The interface might change in future release as `QuantLib` stabilises its own API.

### Author(s)

Dirk Eddelbuettel <edd@debian.org> for the R interface; Khanh Nguyen <nguyen.h.khanh@gmail.com> for the implementation; the QuantLib Group for `QuantLib`

### References

http://quantlib.org for details on `QuantLib`.

## Examples

```
dates <- seq(from=as.Date("2009-04-07"), to=as.Date("2009-04-14"), by=1)
isWeekend("UnitedStates", dates)
isWeekend("UnitedStates/Settlement", dates)       ## same as previous
isWeekend("UnitedStates/NYSE", dates)             ## stocks
isWeekend("UnitedStates/GovernmentBond", dates)  ## bonds
isWeekend("UnitedStates/NERC", dates)             ## energy
```

---

Matlab Certificate of Deposit

*Matlab's certificate of deposit functions*

---

## Description

Accrued interest, price and yield of certificate of deposit functions in Matlab.

## Usage

```
mf.cdai(CouponRate, Settle, Maturity, IssueDate, Basis=6)
mf.cdprice(Yield, CouponRate, Settle, Maturity, IssueDate, Basis=6)
mf.cdyield(Price, CouponRate, Settle, Maturity, IssueDate, Basis=6)
```

## Arguments

| | |
|---|---|
| CouponRate | a numeric, coupon rate of the CD |
| Settle | a Date, settle date of the CD |
| Maturity | a Date, maturity date of the CD |
| IssueDate | a Date, issue date of the CD |
| Yield | a numeric, yield on the CD |
| Price | a numeric, price of the CD |
| Basis | (Optional) a numeric, day counter. Default is 6, or 'Thirty360. See Enum. |

## Value

mf.cdai returns the accrued interest on the CD

mf.cdprice returns a two items list. AccrtInt and Price.

mf.cdyield returns the yield on the CD

## Author(s)

Khanh Nguyen <knguyen@cs.umb.edu>

## References

http://www.mathworks.com/access/helpdesk/help/toolbox/finfixed/f5-6010.
html#f5-8321

### Examples

```
CouponRate <- 0.05
Settle <- as.Date('2002-01-02')
Maturity <- as.Date('2002-03-31')
IssueDate <- as.Date('2001-10-01')

mf.cdai(CouponRate, Settle, Maturity, IssueDate)

Yield <- 0.05225
price <- mf.cdprice(Yield, CouponRate, Settle, Maturity, IssueDate)

mf.cdyield(price$Price, CouponRate, Settle, Maturity, IssueDate)
```

---

```
Matlab Convertible bond pricing
```
*Matlab's convertible bond pricing*

---

### Description

This function mirrors Matlab's cbprice function in Matlab. It price convertible bond.

### Usage

```
mf.cbprice(RiskFreeRate, StaticSpread, Sigma, Price,
                   ConvRatio, NumSteps, IssueDate, Settle,
                   Maturity, CouponRate, Period = 2, Basis = 6,
                   EndMonthRule = 0, DividendType = 0,
                   DividendInfo, CallType = 0, CallInfo, TreeType = 0)
```

### Arguments

RiskFreeRate  a numeric, Annual yield of risk-free bond with the same maturity as the convertible, compounded continuously. (Recommended value is the yield of a risk-free bond with the same maturity as the convertible.

StaticSpread  a numeric, Value of constant spread to the risk free rate. Adding this to the RiskFreeRate produces the issuer's yield, which reflects its credit risk.

Sigma  a numeric, Annual volatility in decimal.

Price  a numeric, Price of asset at settlement or valuation date.

ConvRatio  a numeric, Number of assets convertible to a single bond.

NumSteps  a numeric, Number of steps in binomial tree.

IssueDate  a Date, Issue date of bond.

Settle  a Date, Settlement date of bond.

Maturity  a Date, Maturity date of bond

CouponRate  a numeric, Coupon rate payable per unit of face value.

| Period | (Optional) a numeric, frequency of paymend. Defaul is 2, or 'Semiannual'. |
|---|---|
| Basis | (Optional) a numeric, day counter basis. Default value is 6, or 'Thirty360'. |
| EndMonthRule | (Optional) a numeric. Default value is 0. |
| DividendType | (Optional) 0 = dollar dividend (default). 1 = dividend yield. |
| DividendInfo | (Optional) Two-column data frame of dividend information. First column contains the ex-dividend date corresponding to the amount in the second column. Default = no dividend. |
| CallType | 0 = call on cash price (default). 1 = call on clean price. |
| CallInfo | (Optional) Two-column matrix of call information. First column contains the call dates. Second column contains call prices for every $100 face of bond. A call in the amount of call prices is activated after the corresponding call date. Default = no call feature. |
| TreeType | (Optional) 0 = binomial tree (default). 1 = trinomial tree. |

**Value**

Returns the price of the given convertible bond.

**Author(s)**

Khanh Nguyen <knguyen@cs.umb.edu>

**References**

http://www.mathworks.com/access/helpdesk/help/toolbox/finfixed/f5-6010.
html#f5-8321

**Examples**

```
RiskFreeRate = 0.05
Sigma        = 0.3
ConvRatio    = 1
NumSteps     = 200
IssueDate    = as.Date("2002-01-02")
Settle       = as.Date("2002-01-01")
Maturity     = as.Date("2007-01-02")
CouponRate   = 0.04;
CallInfo     = data.frame(Date=as.Date("2004-01-02"), Amount=110)
CallType     = 1;
TreeType     = 1;

DividendInfo=data.frame(Date=as.Date(character(0)), Amount=numeric(0))

StaticSpread <- 0.005
Price <- 70
mf.cbprice(RiskFreeRate, StaticSpread, Sigma, Price,
ConvRatio, NumSteps, IssueDate, Settle, Maturity, CouponRate, Period=2,
Basis=6, EndMonthRule=0, DividendType=0, DividendInfo,
CallType, CallInfo, TreeType)
```

```
Matlab Zero coupon bond
```
*Matlab's zero coupon bond functions*

### Description

Zero coupon instruments releated functions in Matlab.

### Usage

```
mf.zeroprice(Yield, Settle, Maturity, Period=2, Basis=6, EndMonthRule=0)
mf.zeroyield(Price, Settle, Maturity, Period=2, Basis=6, EndMonthRule=0)
```

### Arguments

| | |
|---|---|
| Yield | a numeric, yield to maturity of instruments |
| Price | a numeric, price of the instrument |
| Settle | a Date, settlement date |
| Maturity | a Date, maturity date |
| Period | (Optional) a numeric, number of quasi-coupon per year. Defaul is 2, or 'Semi-annual'. See Enum |
| Basis | (Optional) a numeric, day counter. Default is 6, or 'Thirty360'. See Enum |
| EndMonthRule | (Optional) a numeric, end of month rule. Default is 0. |

### Value

mf.zeroprice returns the price of a bond given yield

mf.zeroyield returns the yield of the bond given price

### Author(s)

Khanh Nguyen <knguyen@cs.umb.edu>

### References

http://www.mathworks.com/access/helpdesk/help/toolbox/finfixed/f5-6010.
html#f5-8238

## Examples

```
Settle <- as.Date("1993-06-24")
Maturity <- as.Date("1993-11-01")
Period <- 2
Basis <- 2
Yield <- 0.04

mf.zeroprice(Yield, Settle, Maturity, Period, Basis)

Price <- 95
mf.zeroyield(Price, Settle, Maturity, Period, Basis)
```

---

Option                                   *Base class for option price evalution*

---

## Description

This class forms the basis from which the more specific classes are derived.

## Usage

```
## S3 method for class 'Option':
print
## S3 method for class 'Option':
plot
## S3 method for class 'Option':
summary
```

## Arguments

Option          Any option object derived from this base class

## Details

Please see any decent Finance textbook for background reading, and the QuantLib documentation
for details on the QuantLib implementation.

## Value

None, but side effects of displaying content.

## Note

The interface might change in future release as QuantLib stabilises its own API.

## Author(s)

Dirk Eddelbuettel <edd@debian.org> for the R interface; the QuantLib Group for QuantLib

## References

<http://quantlib.org> for details on `QuantLib`.

## See Also

[AmericanOption](),[EuropeanOption](), [BinaryOption]()

## Examples

```
EO<-EuropeanOption("call", strike=100, volatility=0.4, 100, 0.01, 0.03, 0.5)
print(EO)
summary(EO)
```

---

yearFraction            *DayCounter functions from QuantLib*

---

## Description

The `yearFraction` function returns year fraction between two dates given a day counter [Enum]()

## Usage

```
yearFraction(startDates, endDates, dayCounters)
```

## Arguments

startDates    A vector of `Date` type.

endDates     A vector of `Date` type.

dayCounters   A vector of numeric type. See [Enum]()

## Details

The day counters are coming from QuantLib, and the QuantLib documentation should be consulted for details. See [Enum]() and [http://quantlib.org/reference/class_quant_lib_1_1_day_counter.html](http://quantlib.org/reference/class_quant_lib_1_1_day_counter.html)

## Value

A numeric vector contains year fractions between two dates from the input.

## Note

The interface might change in future release as `QuantLib` stabilises its own API.

## Author(s)

Dirk Eddelbuettel <edd@debian.org> for the R interface; Khanh Nguyen <nguyen.h.khanh@gmail.com> for the implementation; the QuantLib Group for `QuantLib`

## References

<http://quantlib.org> for details on `QuantLib`.

## Examples

```
  startDates <- seq(from=as.Date("2009-04-07"), to=as.Date("2009-04-14"),
by=1)
endDates <- seq(from=as.Date("2009-11-07"), to=as.Date("2009-11-14"), by=1)
dayCounters <- c(0,1,2,3,4,5,6,1)
yearFraction(startDates, endDates, dayCounters)
```

---

ZeroCouponBond                 *Zerocoupon bond pricing*

---

## Description

The `ZeroCouponBond` function evaluates a zero-coupon plainly using discount curve. More
specificly, the calculation is done by DiscountingBondEngine from QuantLib. The NPV, clean
price, dirty price, accrued interest, yield and cash flows of the bond is returned. For more detail, see
the source codes in quantlib's test-suite. test-suite/bond.cpp

## Usage

```
## Default S3 method:
ZeroCouponBond(bond, discountCurve, dateparams)

## S3 method for class 'Bond':
plot
## S3 method for class 'Bond':
print
## S3 method for class 'Bond':
summary
```

## Arguments

bond            bond parameters, a named list whose elements are:

|  |  |
|---|---|
| issueDate | a Date, the bond's issue date |
| maturityDate | a Date, the bond's maturity date |
| faceAmount | (Optional) a double, face amount of the bond. Default value is 100. |
| redemption | (Optional) a double, percentage of the initial face amount that will be returned at maturity date. Default value is 100. |

discountCurve

> Can be one of the following:

| | |
|---|---|
| a DiscountCurve | a object of DiscountCurve class For more detail, see example or the discountCurve function |
| A 2 items list | specifies a flat curve in two values "todayDate" and "rate" |
| A 3 items list | specifies three values to construct a DiscountCurve object, "params" , "tsQuotes", "times". For more detail, see example or the discountCurve function |

dateparams    (Optional) a named list, QuantLib's date parameters of the bond.

| | |
|---|---|
| settlementDays | (Optional) a double, settlement days. Default value is 1. |
| calendar | (Optional) a string, either 'us' or 'uk' corresponding to US Goverment Bond calendar and UK Exchange calendar. Default value is 'us'. |
| businessDayConvention | (Optional) a number or string, business day convention. See Enum. Default value is 'Following'. |

> See example below.

## Details

A discount curve is built to calculate the bond value.

Please see any decent Finance textbook for background reading, and the QuantLib documentation for details on the QuantLib implementation.

## Value

The ZeroCouponBond function returns an object of class ZeroCouponBond (which inherits from class Bond). It contains a list with the following components:

| | |
|---|---|
| NPV | net present value of the bond |
| cleanPrice | clean price of the bond |
| dirtyPrice | dirty price of the bond |
| accruedAmount | |
| | accrued amount of the bond |
| yield | yield of the bond |
| cashFlows | cash flows of the bond |

## Note

The interface might change in future release as `QuantLib` stabilises its own API.

## Author(s)

Khanh Nguyen <knguyen@cs.umb.edu> for the inplementation; Dirk Eddelbuettel <edd@debian.org> for the R interface; the QuantLib Group for `QuantLib`

## References

<http://quantlib.org> for details on `QuantLib`.

## Examples

```
# simple call with unnamed parameters
bond <- list(faceAmount=100, issueDate=as.Date("2004-11-30"),
             maturityDate=as.Date("2008-11-30"), redemption=100 )

dateparams <-list(settlementDays=1, calendar="us", businessDayConvention=4)

discountCurve <- list(todayDate=as.Date("2004-11-04"), riskFreeRate=0.03)

ZeroCouponBond(bond, discountCurve, dateparams)

params <- list(tradeDate=as.Date('2002-2-15'),
               settleDate=as.Date('2002-2-19'),
               dt=.25,
               interpWhat="discount",
               interpHow="loglinear")

tsQuotes <- list(d1w  =0.0382,
                 d1m  =0.0372,
                 fut1=96.2875,
                 fut2=96.7875,
                 fut3=96.9875,
                 fut4=96.6875,
                 fut5=96.4875,
                 fut6=96.3875,
                 fut7=96.2875,
                 fut8=96.0875,
                 s3y  =0.0398,
                 s5y  =0.0443,
                 s10y =0.05165,
                 s15y =0.055175)

times <- seq(0,10,.1)
discountCurve <- list(params, tsQuotes, times)

# depreciated
ZeroCouponBond(bond, discountCurve, dateparams)

# construct a curve
```

```
curves <- DiscountCurve(params, tsQuotes, times)
ZeroCouponBond(bond, curves, dateparams)

#construct a flat curve
flatquote <- list(flat=0.04)
flatCurve <- DiscountCurve(params, flatquote, times)
dateparams <-list(settlementDays=1, calendar="us",
                  businessDayConvention="Following")

ZeroCouponBond(bond, flatCurve, dateparams)

#examples with default arguments
curve <- DiscountCurve(params, tsQuotes)
ZeroCouponBond(bond, curve, dateparams)
ZeroCouponBond(bond, curve)

bond <- list(issueDate=as.Date("2004-11-30"),
maturityDate=as.Date("2008-11-30"))
dateparams <-list(settlementDays=1)
ZeroCouponBond(bond, curve, dateparams)
```

---

ZeroPriceByYield       *Zero Coupon Bond Theoretical Price evaluation*

---

### Description

The `ZeroPriceYield` function evaluates a zero-coupon clean price based on its yield.

### Usage

```
## Default S3 method:
ZeroPriceByYield(yield, faceAmount,
                               issueDate, maturityDate,
                               dayCounter=2, frequency=2,
                               compound=0, businessDayConvention=4)
## S3 method for class 'Bond':
plot
## S3 method for class 'Bond':
print
## S3 method for class 'Bond':
summary
```

### Arguments

yield         yield of the bond

faceAmount    face amount of the bond

issueDate     date the bond is issued

maturityDate    maturity date, an R's date type

dayCounter      day count convention. 0 = Actual360(), 1 = Actual365Fixed(), 2 = ActualAc-
                tual(), 3 = Business252(), 4 = OneDayCounter(), 5 = SimpleDayCounter(), all
                other = Thirty360(). For more information, see QuantLib's DayCounter class

frequency       frequency of events,0=NoFrequency, 1=Once, 2=Annual, 3=Semiannual, 4=Ev-
                eryFourthMonth, 5=Quarterly, 6=Bimonthly ,7=Monthly ,8=EveryFourthWeely,9=Biweekly,
                10=Weekly, 11=Daily. For more information, see QuantLib's Frequency class

compound        compounding type. 0=Simple, 1=Compounded, 2=Continuous, all other=SimpleThenCompounded.
                See QuantLib's Compound class

businessDayConvention
                convention used to adjust a date in case it is not a valid business day. See quantlib
                for more detail. 0 = Following, 1 = ModifiedFollowing, 2 = Preceding, 3 =
                ModifiedPreceding, other = Unadjusted

## Value

The ZeroPriceByYield function returns an object of class ZeroPriceByYield (which in-
herits from class Bond). It contains a list with the following components:

yield           yield of the bond

## Note

The interface might change in future release as QuantLib stabilises its own API.

## Author(s)

Khanh Nguyen <knguyen@cs.umb.edu>

## References

http://quantlib.org for details on QuantLib. http://www.mathworks.com/access/
helpdesk/help/toolbox/finfixed/zeroyield.html for more details about this func-
tion

## Examples

ZeroPriceByYield(0.1478, 100, as.Date("1993-6-24"), as.Date("1993-11-1"))

ZeroYield          *Zero Coupon Bond Yield evaluation*

### Description

The `ZeroYield` function evaluations a zero-coupon yield based. See also http://www.mathworks.com/access/helpdesk/help

### Usage

```
## Default S3 method:
ZeroYield(price, faceAmount,
                        issueDate, maturityDate,
                        dayCounter=2, frequency=2,
                        compound=0, businessDayConvention=4)
## S3 method for class 'Bond':
plot
## S3 method for class 'Bond':
print
## S3 method for class 'Bond':
summary
```

### Arguments

| | |
|---|---|
| `price` | price of the bond |
| `faceAmount` | face amount of the bond |
| `issueDate` | date the bond is issued |
| `maturityDate` | maturity date, an R's date type |
| `dayCounter` | day count convention. 0 = Actual360(), 1 = Actual365Fixed(), 2 = ActualActual(), 3 = Business252(), 4 = OneDayCounter(), 5 = SimpleDayCounter(), all other = Thirty360(). For more information, see QuantLib's DayCounter class |
| `frequency` | frequency of events,0=NoFrequency, 1=Once, 2=Annual, 3=Semiannual, 4=EveryFourthMonth, 5=Quarterly, 6=Bimonthly ,7=Monthly ,8=EveryFourthWeely,9=Biweekly, 10=Weekly, 11=Daily. For more information, see QuantLib's Frequency class |
| `compound` | compounding type. 0=Simple, 1=Compounded, 2=Continuous, all other=SimpleThenCompounded. See QuantLib's Compound class |
| `businessDayConvention` | |
| | convention used to adjust a date in case it is not a valid business day. See quantlib for more detail. 0 = Following, 1 = ModifiedFollowing, 2 = Preceding, 3 = ModifiedPreceding, other = Unadjusted |

### Value

The `ZeroYield` function returns an object of class `ZeroYield` (which inherits from class `Bond`). It contains a list with the following components:

| | |
|---|---|
| `yield` | yield of the bond |

**Note**

The interface might change in future release as `QuantLib` stabilises its own API.

**Author(s)**

Khanh Nguyen `<knguyen@cs.umb.edu>`

**References**

`http://quantlib.org` for details on `QuantLib`. `http://www.mathworks.com/access/helpdesk/help/toolbox/finfixed/zeroyield.html` for more details about this function

**Examples**

```
ZeroYield(90, 100, as.Date("1993-6-24"), as.Date("1993-11-1"))
```

# Index