Package: fixedincome (via r-universe)

August 23, 2024

Title Fixed Income Models, Calculations, Data Structures and Instruments

Version 0.0.5

License MIT + file LICENSE

Description Fixed income mathematics made easy. A rich set of functions that helps with calculations of interest rates and fixed income. It has objects that abstract interest rates, compounding factors, day count rules, forward rates and term structure of interest rates. Many interpolation methods and parametric curve models commonly used by practitioners are implemented.

URL https://github.com/wilsonfreitas/R-fixedincome

BugReports https://github.com/wilsonfreitas/R-fixedincome/issues

Depends R (>= 4.0.0),

Imports bizdays (>= 1.0.0), methods, graphics, stats, grDevices, utils, ggplot2, scales

Suggests knitr, rmarkdown, rb3, dplyr, testthat (>= 3.0.0)

Collate ``fixedincome-internal.R'' ``utils.R'' ``term-class.R'' ``daycount-class.R'' ``compounding-class.R'' ``interpolation-class.R'' ``spotrate-class.R'' ``spotratecurve-class.R'' ``spotratecurve-interpolation.R'' ``spotratecurve-interpolation-nelsonsiegel.R'' ``plot.R'' ``autoplot.R'' ``forwardrate-class.R'' ``compound-method.R'' ``hidden.R'' ``zzz.R''

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.1

Encoding UTF-8

LazyData true

Config/testthat/edition 3

VignetteBuilder knitr

Repository https://wilsonfreitas.r-universe.dev

RemoteUrl https://github.com/wilsonfreitas/r-fixedincome

RemoteRef HEAD

RemoteSha e6a94eb6d320dc0245e5c7188ba690bc4b37b120

Contents

fixedincome-package *Fixed income models, calculations and data structures*

Description

The fixedincome package brings a set of funtions that helps with the mathematics of interest rates and fixed income. It handles the interest rates and term structures of interest rates as objects and provides many methods to tackle specific issues like computing discount factors and forward rates, interpolate term structures, fit curve models and so much more. This package also supports methods and models commonly used by practitioners to do fixed income calculations.

Author(s)

Wilson Freitas <wilson.freitas@gmail.com>

References

Frank Fabozzi. Fixed Income Mathematics, Wiley, 1994.

Bruce Tuckman. Fixed Income Securities, Wiley, 1994.

as.forwardrate *Coerce objects to ForwardRate*

Description

A ForwardRate object can be created from a SpotRate object and a SpotRateCurve.

Usage

```
as.forwardrate(x, ...)

## S3 method for class 'SpotRate'
as.forwardrate(x, terms, refdate = NULL, ...)

## S3 method for class 'SpotRateCurve'
as.forwardrate(x, t1 = NULL, t2 = NULL, ...)
```

Arguments

x	a SpotRate or a SpotRateCurve object.
...	additional arguments
terms	a numeric with positive values representing terms or a Term object.
refdate	the curve reference date.
t1	initial term
t2	final term

Value

A ForwardRate object created from another object, SpotRate or SpotRateCurve.

---

as.spotrate *Coerce to SpotRate*

Description

Coerce character objects to SpotRate class

Usage

```
as.spotrate(x, ...)

## S4 method for signature 'character'
as.spotrate(x, simplify = TRUE)

## S4 method for signature 'SpotRateCurve'
as.spotrate(x, ...)

## S4 method for signature 'ForwardRate'
as.spotrate(x, ...)
```

Arguments

x	a character with SpotRate specification.
...	additional arguments
simplify	a boolean indicating whether to simplify SpotRate creation or not. Defaults to TRUE.

Details

The character representation of a SpotRate is as follows:

```
"RATE COMPOUNDING DAYCOUNT CALENDAR"
```

where:

- RATE is a numeric value
- COMPOUNDING is one of the following: simple, discrete, continuous
- DAYCOUNT is a valid day count rule, pex. business/252, see Daycount.
- CALENDAR is the name of a bizdays calendar.

simplify check if compounding, daycount and calendar are the same for all given characters. If it is true the returned object is a SpotRate otherwise a list with SpotRate objects is returned.

Value

A SpotRate object created from a string.

Examples

```
as.spotrate(c(
  "0.06 simple actual/365 actual",
  "0.11 discrete business/252 actual"
))
```

---

as.spotratecurve *Coerce objects to spotratecurve*

Description

A SpotRateCurve can be created from a ForwardRate object.

Usage

```
as.spotratecurve(x, ...)

## S3 method for class 'ForwardRate'
as.spotratecurve(x, refdate = Sys.Date(), ...)
```

Arguments

x	a ForwardRate object.
...	additional arguments
refdate	the curve reference date.

Value

A SpotRateCurve object create from another object.

---

as.term *Coerce a character to a Term*

Description

as.term coerces a character vector to a Term object.

Usage

```
as.term(x, ...)
```

Arguments

x	a character to be coerced to a Term.
...	additional arguments. Currently unused.

Details

The string representation of the Term class follows the layout:

```
NUMBER UNITS
```

where units is one of: days, months, years.

Value

A Term object created from a string.

Examples

```
t <- as.term("6 months")
```

---

compound *Compound method*

Description

Computes the compounding (and discount) factor for spot rates and curves.

Usage

```
compound(x, t, val, ...)

discount(x, t, val, ...)
```

Arguments

x	can be a Compounding, a SpotRate, a SpotRateCurve, a ForwardRate and a character representing a Compounding.
t	represents the term to compound. Can be a numeric, a Term, a Date or even missing. See Details.
val	is the value of the spot rate to be compounded in the given term. Can be a numeric, a Date or missing. See Details.
...	additional arguments.

Details

For Compounding classes the arguments t and val must be provided.

For a SpotRate class, if the t argument is numeric, representing the term to be compounded, the argument val must be a character with the units of the Term class. If otherwise t is a Term object, val is missing.

For SpotRateCurve and ForwardRate classes, that already have terms associated, t and val are missing.

discount() method is the inverse of compound: 1 / compound().

Value

A numeric value that represents the compounding factor for the given spot rate.

Examples

```
compound("simple", 2, 0.05)
compound("discrete", 2, 0.05)
compound("continuous", 2, 0.05)

spr <- spotrate(0.06, "simple", "actual/365", "actual")
compound(spr, 10, "days")
discount(spr, 10, "days")
t <- term(10, "days")
compound(spr, t)
discount(spr, t)
d1 <- Sys.Date()
d2 <- Sys.Date() + 10
compound(spr, d1, d2)
discount(spr, d1, d2)

terms <- c(1, 11, 26, 27, 28)
rates <- c(0.0719, 0.056, 0.0674, 0.0687, 0.07)
curve <- spotratecurve(rates, terms, "discrete", "actual/365", "actual")
compound(curve)
discount(curve)
```

---

compounding *Create Compounding class*

Description

compound() creates a Compounding object in one of its subclasses: Simple, Discrete, Continuous.

Usage

```
compounding(x = c("simple", "discrete", "continuous"))
```

Arguments

x	a character with the name of compounding regime: simple, discrete, continuous

Details

A Compounding object can be instanciated with the compounding function, passing a string with the name of one of the compounding regimes: simple, discrete, continuous.

Value

A subclass of Compounding object.

Examples

```
compounding("simple")
compounding("discrete")
compounding("continuous")

comp <- compounding("discrete")
compound(comp, 0.06, 2) # equals (1 + 0.06) ^ 2 = 1.1236
implied_rate(comp, 1.1236, 2) # equals 0.06
```

Compounding-class *Compounding class*

Description

The Compounding class abstracts the compounding regime used to discount or compound a spot rate.

Details

There are 3 compoundings:

- simple for simple interest rate compounding

$$1 + rt$$

- discrete for compounded interest rate compounding

$$(1 + r)^t$$

- continuous for continuous interest rate compounding

$$exp(rt)$$

The Compounding class has 2 methods:

- compound to compound the spot rate for a given term.
- rates to compute the implied rate for a compound factor in a given term.

datasets *Datasets*

Description

Interest rate datasets

Details

`ZeroCurveBRL` Brazil's zero curve

daycount *Create Daycount class*

Description

`daycount` creates a `Daycount` object. It accepts the following daycount rules: `actual/365`, `actual/360`, `business/252`.

Usage

```
daycount(x, ...)
```

Arguments

`x`	a character representing a daycount rule, like: `business/252`, `actual/365`, `actual/360`, ...
`...`	additional arguments. Currently unused.

Value

A `Daycount` object.

Examples

```
dc <- daycount("actual/360")
```

Daycount-class *Daycount class*

Description

Daycount class helps adjusting the terms to compound interest rates. With annual rates it is necessary to convert periods of days or months to years units. The day count convention helps with that by defining the number of days of one year. Together with a calendar it defines the way the wordays are counted between two dates.

Details

Common day count rules are: actual/365, actual/360, business/252, 30/360, ...

dib *Days in base for Daycount*

Description

dib returns the days in base, that is the number of days used to define one year.

Usage

```
dib(x)
```

Arguments

x	a Daycount object.

Details

The method dib returns the days in base for a daycount convention. Since we work with annual rates the days in base define the amount of days in a year used in the convention.

Value

A numeric with daycount's days in base, the number of days in a year used in the convention.

Examples

```
dc <- daycount("actual/360")
dib(dc)
```

diff,Term-method *Calculate lagged differences of Term objects*

Description

diff returns a Term vector with lagged differences.

Usage

```
## S4 method for signature 'Term'
diff(x, lag = 1, ..., fill = NULL)
```

Arguments

x	a Term object.
lag	a numerix indicating which lag to use.
...	additional arguments. Currently unused.
fill	a numeric value (or NA) to fill the empty created by applying diff to a Term object.

Value

A new Term object with lagged differences of the given Term object.

Examples

```
t <- term(1:10, "months")
diff(t)
```

fit_interpolation *Fit parametric interpolation functions*

Description

Fits parametric interpolation functions like NelsonSiegel or NelsonSiegelSvensson.

Usage

```
fit_interpolation(object, x, ...)
```

Arguments

object	a Interpolation object with initial parameters set.
x	a SpotRateCurve object.
...	additional arguments. Currently unused.

Value

A Interpolation object.

Examples

```
terms <- c(1, 11, 26, 27, 28)
rates <- c(0.0719, 0.056, 0.0674, 0.0687, 0.07)
curve <- spotratecurve(rates, terms, "discrete", "actual/365", "actual")
fit_interpolation(interp_nelsonsiegel(0.1, 0.01, 0.01, 0.01), curve)
```

---

forwardrate *Create a ForwardRate object*

Description

forwardrate() creates a ForwardRate object.

Usage

```
forwardrate(x, ...)

## S3 method for class 'numeric'
forwardrate(
  x,
  terms,
  compounding,
  daycount,
  calendar,
  .copyfrom = NULL,
  refdate = NULL,
  ...
)

## S3 method for class 'SpotRateCurve'
forwardrate(x, t1 = NULL, t2 = NULL, ...)
```

Arguments

x	a numeric or a SpotRateCurve object.
...	additional arguments.
terms	a numeric vector with positive values representing terms of the forward rates.
compounding	a character with the compouning name.
daycount	a character representing the daycount.
calendar	a calendar object.
.copyfrom	a SpotRate object that is used as reference to build the SpotRateCurve object.

refdate	the curve reference date.
t1	initial term
t2	final term

Value

A ForwardRate object.

The arguments t1 and t2 define initial and final term used to extract a ForwardRate from a SpotRateCurve.

ForwardRate-class *ForwardRate class*

Description

ForwardRate class abstracts a forward rate. It has an additional term, that reffers to the forward period used to compute the forward rate.

---

ggplot2-support *ggplot2 plotting functions*

Description

Functions to plot fixedincome objects using ggplot2 package and its grammar of graphics.

Usage

```
## S3 method for class 'SpotRateCurve'
autoplot(
  object,
  ...,
  curve.name = NULL,
  curve.geom = c("line", "point"),
  curve.interpolation = FALSE,
  curve.x.axis = c("dates", "terms")
)

## S3 method for class 'SpotRateCurve'
autolayer(
  object,
  ...,
  curve.name = NULL,
  curve.geom = c("line", "point"),
  curve.interpolation = FALSE,
  curve.x.axis = c("dates", "terms")
```

```
)

## S3 method for class 'ForwardRate'
autolayer(
  object,
  ...,
  curve.name = NULL,
  curve.geom = c("step", "line", "point"),
  curve.x.axis = c("dates", "terms")
)
```

Arguments

`object`	SpotRateCurve or ForwardRate objects
`...`	additional arguments passed to ggplot2 geom_* functions
`curve.name`	Curve's name
`curve.geom`	Curve geom used: `line`, `point`, `step`
`curve.interpolation`	logical indicating to use curve interpolation in the plot. Defaults to `FALSE`.
`curve.x.axis`	x axis can be presented with a numeric scale representing business days (`terms`) or dates (`dates`). Defaults to `dates`.

`ggspotratecurveplot` *Fancy ggplot for SpotRateCurve object*

Description

Fancy ggplot for SpotRateCurve object with custom axis, title

Usage

```
ggspotratecurveplot(
  curve,
  title = NULL,
  subtitle = NULL,
  caption = NULL,
  curve.name = NULL,
  curve.interpolation = FALSE,
  curve.x.axis = c("dates", "terms"),
  ...
)
```

Arguments

curve	SpotRateCurve object
title	plot title
subtitle	plot subtitle
caption	plot caption
curve.name	Curve's name, if not provided curve's refdate is used.
curve.interpolation	logical indicating to use daily interpolation instead of curve points. Defaults to FALSE.
curve.x.axis	x axis can be presented with a numeric scale representing business days (terms) or dates (dates). Defaults to dates.
...	additional arguments (not used)

Examples

```
## Not run:
terms <- c(1, 11, 26, 27, 28)
rates <- c(0.0719, 0.056, 0.0674, 0.0687, 0.07)
curve <- spotratecurve(rates, terms, "discrete", "actual/365", "actual")
ggspotratecurveplot(curve)

## End(Not run)
```

---

implied_rate *Implied rates*

Description

Computes implied rates to compounding factors.

Usage

```
implied_rate(x, t, val, ...)
```

Arguments

x	a Compounding object or a character with the compounding name.
t	a numeric representing the term.
val	a numeric representing the compounding factor.
...	additional arguments. Currently unused.

Details

If the x argument is a character with a valid compounding name (simple, discrete, continuous) the function instanciates a Compounding object and then computes the implied rate for the given compounding values and terms.

Value

A numeric value that represents a spot rate.

Examples

```
implied_rate("simple", 2, 1.1)
implied_rate("discrete", 2, 1.1025)
implied_rate("continuous", 2, 1.105170918)

comp <- compounding("discrete")
compound(comp, 0.06, 2) # equals (1 + 0.06) ^ 2 = 1.1236
implied_rate(comp, 1.1236, 2) # equals 0.06
```

---

interpolation *Set/Get interpolation to SpotRateCurve*

Description

Sets and gets interpolation method to the SpotRateCurve.

Usage

```
interpolation(x, ...)

interpolation(x) <- value
```

Arguments

x	a SpotRateCurve object.
...	additional arguments. Currently unused.
value	a Interpolation object.

Value

A Interpolatin object.

Examples

```
terms <- c(1, 11, 26, 27, 28)
rates <- c(0.0719, 0.056, 0.0674, 0.0687, 0.07)
curve <- spotratecurve(rates, terms, "discrete", "actual/365", "actual")
interpolation(curve) <- interp_flatforward()
interpolation(curve)
```

Interpolation-class *Interpolation classes*

Description

Classes that implement interpolation methods to be used with `SpotRateCurve` objects.

Details

- FlatForward
- Linear
- LogLinear
- NaturalSpline
- HermiteSpline
- MonotoneSpline
- NelsonSiegel
- NelsonSiegelSvensson

Every class that implement a interpolation method inherits the `Interpolation` class.

interpolation-constructor *Create Interpolation objects*

Description

Functions to create intepolation objects.

Usage

```
interp_flatforward()

interp_linear()

interp_loglinear()

interp_naturalspline()

interp_hermitespline()

interp_monotonespline()

interp_nelsonsiegel(beta1, beta2, beta3, lambda1)

interp_nelsonsiegelsvensson(beta1, beta2, beta3, beta4, lambda1, lambda2)
```

Arguments

beta1	a single numeric
beta2	a single numeric
beta3	a single numeric
lambda1	a single numeric
beta4	a single numeric
lambda2	a single numeric

Details

interp_flatforward creates a FlatForward interpolation object.

interp_linear creates a Linear interpolation object.

interp_loglinear creates a LogLinear interpolation object.

interp_naturalspline creates a NaturalSpline interpolation object.

interp_hermitespline creates a HermiteSpline interpolation object.

interp_monotonespline creates a MonotoneSpline interpolation object.

interp_nelsonsiegel creates a NelsonSiegel interpolation object. The arguments beta1, beta2, beta3, lambda1 are the paremeters of the Nelson-Siegel model for term structure.

interp_nelsonsiegelsvensson creates a NelsonSiegelSvensson interpolation object. The arguments beta1, beta2, beta3, beta4, lambda1, lambda2 are the paremeters of Svensson's extension to Nelson-Siegel the model for term structure.

Value

An Interpolation object. That object knows the interpolation method but doesn't have the data points. When the Interpolation is set to the curve with interpolation<- the interpolation engine is properly configured.

References

Charles R. Nelson and Andrew F. Siegel (1987), The Journal of Business

Lars E.O. Svensson (1994), National Bureau of Economic Research

Examples

```
terms <- c(1, 11, 26, 27, 28)
rates <- c(0.0719, 0.056, 0.0674, 0.0687, 0.07)

curve <- spotratecurve(rates, terms, "discrete", "actual/365", "actual")

interpolation(curve) <- interp_flatforward()

curve[[1:10]]
```

interpolation_error *Interpolation error*

Description

Computes interpolation error as the root mean square error of differences between interpolated terms and SpotRateCurve values.

Usage

```
interpolation_error(x, ...)
```

Arguments

x	a SpotRateCurve object.
...	additional arguments. Currently unused.

The curve must have a interpolation set to compute the interpolation error. This is useful to evaluate parametric methods like NelsonSiegel and NelsonSiegelSvensson.

Value

A numeric value with the root mean squared error between the curve data point and interpolated points.

Examples

```
terms <- c(1, 11, 26, 27, 28)
rates <- c(0.0719, 0.056, 0.0674, 0.0687, 0.07)
curve <- spotratecurve(rates, terms, "discrete", "actual/365", "actual")
interpolation(curve) <- interp_nelsonsiegel(
  0.1229, -0.0606, 0.1004, 1.9174
)
interpolation_error(curve)
```

maturities *Get SpotRateCurve terms as Date objects*

Description

Compute the SpotRateCurve terms as Date objects, according to the curve's reference date.

Usage

```
maturities(x)
```

Arguments

x	a SpotRateCurve object.

Value

A vector of `Date` objects that represent the curve's `terms` and using curve's `refdate` as a starting point.

Examples

```
terms <- c(1, 11, 26, 27, 28)
rates <- c(0.0719, 0.056, 0.0674, 0.0687, 0.07)
curve <- spotratecurve(rates, terms, "discrete", "actual/365", "actual")
maturities(curve)
```

---

parameters *Get parameters of the interpolation models*

Description

Gets parameters of parametric interpolation models like NelsonSiegel and NelsonSiegelSvensson.

Usage

```
parameters(x, ...)
```

Arguments

x	a Interpolation object.
...	additional arguments. Currently unused.

Value

A named vector with parameters of the models.

Examples

```
terms <- c(1, 11, 26, 27, 28)
rates <- c(0.0719, 0.056, 0.0674, 0.0687, 0.07)
curve <- spotratecurve(rates, terms, "discrete", "actual/365", "actual")
model <- fit_interpolation(interp_nelsonsiegel(0.1, 0.01, 0.01, 0.01), curve)
parameters(model)
```

prepare_interpolation *Create the interpolation function*

Description

Creates the interpolation function to a SpotRateCurve object.

Usage

```
prepare_interpolation(object, x, ...)
```

Arguments

object	a Interpolation object.
x	a SpotRateCurve object.
...	additional arguments. Currently unused.
	This method is used internally when the interpolation is set to a curve. It uses the current state of the curve to build the interpolation function. This is similar to call approxfun and splinefun to create functions that perform interpolation of the given data points.
	This method shouldn't be directly called, it is for internal use only.

Value

A Interpolation object with the slot func properly defined. This slot is set with a function (closure) that executes the interpolation method.

Examples

```
terms <- c(1, 11, 26, 27, 28)
rates <- c(0.0719, 0.056, 0.0674, 0.0687, 0.07)
curve <- spotratecurve(rates, terms, "discrete", "actual/365", "actual")
prepare_interpolation(interp_flatforward(), curve)
```

shift *Shift vectors*

Description

Element wise shift of vectors by k positions.

Usage

```
shift(x, k = 1, ..., fill = NA)
```

Arguments

x	a vector object.
k	a numeric with the number of elements to shift the Term vector
...	additional arguments. Currently unused.
fill	a numeric value (or NA) to fill the empty created by shifting a vector object.

Value

A shifted vector object of the same type of provided object.

Examples

```
shift(1:10, fill = 0)

t <- term(1:10, "months")
shift(t)
```

---

spotrate *Create SpotRate objects*

Description

`spotrate()` function creates `SpotRate` objects.

Usage

```
spotrate(x, compounding, daycount, calendar, .copyfrom = NULL)
```

Arguments

x	a numeric vector representing spot rate values.
compounding	a `Compounding` object.
daycount	a `Daycount` object.
calendar	a `bizdays` calendar.
.copyfrom	a `SpotRate` object used as reference to copy attributes.

Value

A `SpotRate` object.

Examples

```
spotrate(0.06, "continuous", "actual/365", "actual")
spotrate(c(0.06, 0.07, 0.08), "continuous", "actual/365", "actual")
```

SpotRate-class *SpotRate class*

Description

The SpotRate class abstracts the interst rate and has methods to handle many calculations on it.

Details

The SpotRate class fully specifies spot rates. It has:

- the spot rate values which are numeric values representing the rate.
- the compounding regime that specifies how to compound the spot rate. This is a Compounding object.
- the daycount rule to compute the compounding periods right adjusted to the spot rate frequency.
- the calendar according to which the number of days are counted.

The SpotRate class is a numeric, that represents the interest rate and that has the slots: compounding, daycount and calendar.

For example, an annual simple interest rate of 6%, that compounds in calendar days, is defined as follows:

```
sr_simple <- spotrate(0.06, "simple", "actual/360", "actual")
sr_simple
#> [1] "0.06 simple actual/360 actual"
```

actual/360 is the daycount rule and actual is the calendar.

Differently, an annual compound interest rate of 10%, that compounds in business days according to calendar Brazil/ANBIMA is

```
sr_disc <- spotrate(0.1, "discrete", "business/252", "Brazil/ANBIMA")
sr_disc
#> [1] "0.1 discrete business/252 Brazil/ANBIMA"
```

The calendar slot is a bizdays calendar.

An $100,000 investment in an instrument that pays that interst rate for 5 years has the future value.

```
100000 * compound(sr_disc, term(5, "years"))
#> [1] 161051
```

for the simple interest rate we have

```
100000 * compound(sr_simple, term(5, "years"))
#> [1] 130000
```

SpotRate objects can be created with vectors

```
rates <- c(1.69, 0.16, 0.07, 0.72, 0.10, 1.60, 0.18, 1.56, 0.60, 1.69)
sr_vec <- spotrate(rates, "discrete", "business/252", "Brazil/ANBIMA")
sr_vec
#>  [1] "1.69 discrete business/252 Brazil/ANBIMA"
#>  [2] "0.16 discrete business/252 Brazil/ANBIMA"
#>  [3] "0.07 discrete business/252 Brazil/ANBIMA"
#>  [4] "0.72 discrete business/252 Brazil/ANBIMA"
#>  [5] "0.10 discrete business/252 Brazil/ANBIMA"
#>  [6] "1.60 discrete business/252 Brazil/ANBIMA"
#>  [7] "0.18 discrete business/252 Brazil/ANBIMA"
#>  [8] "1.56 discrete business/252 Brazil/ANBIMA"
#>  [9] "0.60 discrete business/252 Brazil/ANBIMA"
#> [10] "1.69 discrete business/252 Brazil/ANBIMA"
```

and can be put into a `data.frame`

```
data.frame(spot_rate = sr_vec)
#>                                   spot_rate
#> 1  1.69 discrete business/252 Brazil/ANBIMA
#> 2  0.16 discrete business/252 Brazil/ANBIMA
#> 3  0.07 discrete business/252 Brazil/ANBIMA
#> 4  0.72 discrete business/252 Brazil/ANBIMA
#> 5  0.10 discrete business/252 Brazil/ANBIMA
#> 6  1.60 discrete business/252 Brazil/ANBIMA
#> 7  0.18 discrete business/252 Brazil/ANBIMA
#> 8  1.56 discrete business/252 Brazil/ANBIMA
#> 9  0.60 discrete business/252 Brazil/ANBIMA
#> 10 1.69 discrete business/252 Brazil/ANBIMA
```

once in a `data.frame`, dplyr verbs can be used to manipulate it.

```
require(dplyr, warn.conflicts = FALSE)
#> Loading required package: dplyr

data.frame(spot_rate = sr_vec) |>
   mutate(comp = compound(spot_rate, term(5, "months")))
#>                                   spot_rate     comp
#> 1  1.69 discrete business/252 Brazil/ANBIMA 1.510301
#> 2  0.16 discrete business/252 Brazil/ANBIMA 1.063794
#> 3  0.07 discrete business/252 Brazil/ANBIMA 1.028592
#> 4  0.72 discrete business/252 Brazil/ANBIMA 1.253536
#> 5  0.10 discrete business/252 Brazil/ANBIMA 1.040512
#> 6  1.60 discrete business/252 Brazil/ANBIMA 1.489037
#> 7  0.18 discrete business/252 Brazil/ANBIMA 1.071398
#> 8  1.56 discrete business/252 Brazil/ANBIMA 1.479449
#> 9  0.60 discrete business/252 Brazil/ANBIMA 1.216326
#> 10 1.69 discrete business/252 Brazil/ANBIMA 1.510301
```

SpotRate is `numeric`, so it executes arithmetic and comparison operations with `numeric` objects.

```
data.frame(spot_rate = sr_vec) |>
   mutate(
     new_spot_rate = spot_rate + 0.02,
     check_gt_1pp = spot_rate > 0.01,
     check_gt_nsr = spot_rate > new_spot_rate
   )
#>                                  spot_rate
#> 1  1.69 discrete business/252 Brazil/ANBIMA
#> 2  0.16 discrete business/252 Brazil/ANBIMA
#> 3  0.07 discrete business/252 Brazil/ANBIMA
#> 4  0.72 discrete business/252 Brazil/ANBIMA
#> 5  0.10 discrete business/252 Brazil/ANBIMA
#> 6  1.60 discrete business/252 Brazil/ANBIMA
#> 7  0.18 discrete business/252 Brazil/ANBIMA
#> 8  1.56 discrete business/252 Brazil/ANBIMA
#> 9  0.60 discrete business/252 Brazil/ANBIMA
#> 10 1.69 discrete business/252 Brazil/ANBIMA
#>                                new_spot_rate check_gt_1pp check_gt_nsr
#> 1  1.71 discrete business/252 Brazil/ANBIMA         TRUE        FALSE
#> 2  0.18 discrete business/252 Brazil/ANBIMA         TRUE        FALSE
#> 3  0.09 discrete business/252 Brazil/ANBIMA         TRUE        FALSE
#> 4  0.74 discrete business/252 Brazil/ANBIMA         TRUE        FALSE
#> 5  0.12 discrete business/252 Brazil/ANBIMA         TRUE        FALSE
#> 6  1.62 discrete business/252 Brazil/ANBIMA         TRUE        FALSE
#> 7  0.20 discrete business/252 Brazil/ANBIMA         TRUE        FALSE
#> 8  1.58 discrete business/252 Brazil/ANBIMA         TRUE        FALSE
#> 9  0.62 discrete business/252 Brazil/ANBIMA         TRUE        FALSE
#> 10 1.71 discrete business/252 Brazil/ANBIMA         TRUE        FALSE
```

SpotRate vectors also are created with the concatenation function `c`.

```
c(sr_disc, 0.1, 0.13, 0.14, 0.15)
#> [1] "0.10 discrete business/252 Brazil/ANBIMA"
#> [2] "0.10 discrete business/252 Brazil/ANBIMA"
#> [3] "0.13 discrete business/252 Brazil/ANBIMA"
#> [4] "0.14 discrete business/252 Brazil/ANBIMA"
#> [5] "0.15 discrete business/252 Brazil/ANBIMA"
```

Furtherly, all indexing operations of numeric objects are supported by SpotRate objects.

Invalid Operations:

Operations involving SpotRate objects with different `compounding`, `daycount` or `calendar`, raise errors.

This happens with the following operations:

- Compare: >, <, <=, >=
- Arithmetic: +, -, *, /

- Concatenation: c

```
try(sr_simple + sr_disc)
#> Error in stop_if_spotrate_slots_differ(e1, e2, "SpotRate objects have different slots") :
#>   SpotRate objects have different slots
try(sr_simple > sr_disc)
#> Error in stop_if_spotrate_slots_differ(e1, e2, "SpotRate objects have different slots") :
#>   SpotRate objects have different slots
try(c(sr_simple, sr_disc))
#> Error in stop_if_spotrate_slots_differ(x, values_, "SpotRate objects have different slots") :
#>   SpotRate objects have different slots
```

Note

The SpotRate objects are annual rates.

---

spotrate-compare-method

SpotRate comparison operations

Description

Comparison operations with SpotRate class SpotRate objects can be compared among themselves or with numeric variables.

Usage

```
## S4 method for signature 'SpotRate,SpotRate'
e1 >= e2

## S4 method for signature 'SpotRate,SpotRate'
e1 <= e2

## S4 method for signature 'SpotRate,SpotRate'
e1 < e2

## S4 method for signature 'SpotRate,SpotRate'
e1 > e2

## S4 method for signature 'SpotRate,SpotRate'
e1 == e2

## S4 method for signature 'SpotRate,SpotRate'
e1 != e2

## S4 method for signature 'SpotRate,numeric'
Compare(e1, e2)
```

```
## S4 method for signature 'numeric,SpotRate'
Compare(e1, e2)
```

Arguments

e1	a SpotRate object or a numeric
e2	a SpotRate object or a numeric

Value

A boolean logical object. The comparison with SpotRate objects only takes all fields into account. Comparing SpotRate against numeric values is equivalent to coerce the SpotRate object to numeric execute the operation, this is a syntax sugar for a shortcut that is commonly applied.

Examples

```
spr <- as.spotrate("0.06 simple actual/365 actual")
spr == 0.06
spr != 0.05
spr > 0.05
spr < 0.1
spr >= 0.05
spr <= 0.1

spr1 <- spotrate(0.06, "simple", "actual/365", "actual")
spr2 <- spotrate(0.02, "simple", "actual/365", "actual")
spr1 == spr2
spr1 != spr2
spr1 > spr2
spr1 < spr2
spr1 >= spr2
spr1 <= spr2

# compare spotrate with different slots
spr2 <- spotrate(0.06, "discrete", "actual/365", "actual")
spr1 == spr2
spr1 != spr2
try(spr1 > spr2)
try(spr1 < spr2)
try(spr1 >= spr2)
try(spr1 <= spr2)
```

---

spotratecurve *Create a SpotRateCurve object*

Description

`spotratecurve()` S3 method createas a SpotRateCurve object. It is dispatched for numeric values, that represent spot rates and for SpotRate objects.

Usage

```
spotratecurve(x, terms, ..., refdate = Sys.Date())

## S3 method for class 'numeric'
spotratecurve(
  x,
  terms,
  compounding,
  daycount,
  calendar,
  refdate = Sys.Date(),
  .copyfrom = NULL,
  ...
)

## S3 method for class 'SpotRate'
spotratecurve(x, terms, refdate = Sys.Date(), .copyfrom = NULL, ...)
```

Arguments

`x`	a numeric representing a spot rate value or a SpotRate object.
`terms`	a numeric vector with positive values representing the days of the term structure.
`...`	additional arguments
`refdate`	the curve reference date.
`compounding`	a character with the compouning name.
`daycount`	a character representing the daycount.
`calendar`	a calendar object.
`.copyfrom`	a SpotRate object that is used as reference to build the SpotRateCurve object.

Value

A `SpotRateCurve` object.

Examples

```
terms <- c(1, 11, 26, 27, 28)
rates <- c(0.0719, 0.056, 0.0674, 0.0687, 0.07)

curve <- spotratecurve(rates, terms, "discrete", "actual/365", "actual")

# access the term 11 days
curve[[11]]
```

```
# access the second element
curve[2]
```

`SpotRateCurve-class` *SpotRateCurve class*

Description

The SpotRateCurve class abstracts a term structure of SpotRate objects. The SpotRateCurve has a reference date (`refdate` slot), that is a mark to market date. The SpotRates are indexed to future dates according to its reference date and these future dates represent the terms of the SpotRateCurve.

Details

Once the SpotRateCurve object is built, any SpotRate can be accessed using indexing operations: `[]` positional indexing, `[[]]` term indexing.

The SpotRateCurve inherits SpotRate class and has three slots: terms that is a Term object, refdate and interpolation that defines the method used to interpolate the curve.

`spotratecurve-helpers` *SpotRateCurve helpers*

Description

Helpers methods that return parts of a SpotRateCurve object according to a given term.

Usage

```
first(x, t)

last(x, t)

closest(x, t)
```

Arguments

`x`	a SpotRateCurve object.
`t`	a Term object. `first` filters the first elements of the SpotRateCurve according to the given term. `last` filters the last elements of the SpotRateCurve according to the given term. `closest` selects the element of the SpotRateCurve that is the closest to the given term.

Value

A SpotRateCurve object that is a subset of the given curve. The elements returned are select according to the operation executed.

Examples

```
terms <- c(1, 11, 26, 27, 28)
rates <- c(0.0719, 0.056, 0.0674, 0.0687, 0.07)
curve <- spotratecurve(rates, terms, "discrete", "actual/365", "actual")
first(curve, "10 days")
last(curve, "10 days")
closest(curve, "10 days")
```

---

term *Create Term class*

Description

term() creates a Term object.

Usage

```
term(x, ...)

## S3 method for class 'numeric'
term(x, units = "days", ...)

## S3 method for class 'Term'
term(x, ...)

## S3 method for class 'Date'
term(x, end_date, calendar, ...)
```

Arguments

x	can be a numeric value representing the time period, a Term object, or the initial date for a period between two dates.
...	additional arguments
units	one of the valid units: days, monts, years.
end_date	the final date for a period between two dates.
calendar	the calendar used to compute the amount of days for a period between two dates.

Value

A Term object.

Examples

```
term(6, "months")
if (require("bizdays")) {
  term(as.Date("2022-02-02"), as.Date("2022-02-23"), "Brazil/ANBIMA")
}
```

`Term-class` *Term class*

Description

It is the time interval used in calculations with interest rates. The Term class represents the period used to discount or compound a spot rate.

Details

The Term object is defined by its numeric value and its unit, that can be `"days"`, `"months"` or `"years"`. For example:

```
term(6, "months")
#> [1] "6 months"
```

It represents a period of 6 months. The Term object can also be created from a string representation of a Term.

```
as.term("6 months")
#> [1] "6 months"
```

Since the Term object inherits from a `numeric`, it inherits all numeric operations. Numeric values can be summed or subtracted from a Term object numeric part.

```
term(1, "days") + 1
#> [1] "2 days"
```

Arithmetic and comparison operations between Term object are not implemented, so these operations raise an error.

```
try(term(1, "days") + term(2 , "days"))
#> Error in term(1, "days") + term(2, "days") : Not implemented
```

DateRangeTerm objects:

The DateRangeTerm class inherits Term and defines start and end dates and a calendar to count the amount of working days between these two dates. This is a Term between two dates.

```
term(Sys.Date() - 5, Sys.Date(), "Brazil/ANBIMA")
#> [1] "2 days"
```

In financial markets it is fairly usual to evaluate interest rates between two dates.

`term-conversion` *Convert Term in different units*

Description

`toyears`, `tomonths` and `todays` functions convert Term objects according to Daycount.

Usage

```
toyears(x, t)

tomonths(x, t)

todays(x, t)
```

Arguments

`x`	a Daycount object.
`t`	a Term object.

Details

`toyears` returns the given Term in years units. `tomonths` returns the given Term in months units. `todays` returns the given Term in days units.

Value

A Term object converted to the units accordingly the used function.

Examples

```
dc <- daycount("actual/360")
t <- term(10, "months")
toyears(dc, t)
tomonths(dc, t)
todays(dc, t)
```

Index