Package 'creditr'

July 22, 2025

Version 0.6.2

Date 2024-12-15

Title Credit Default Swaps

Depends R (>= 3.1.0)

Imports utils, quantmod, devtools, methods, Rcpp (>= 0.10.6), RCurl, XML, zoo, xts

LinkingTo Rcpp

Suggests testthat

License file LICENSE

URL https://github.com/yanrong-stacy-song/creditr

Description Price credit default swaps using 'C' code from the International Swaps and Derivatives Association CDS Standard Model. See <https://www.cdsmodel.com/cdsmodel/documentation.html> for more information about the model and <https://www.cdsmodel.com/cdsmodel/cds-disclaimer.html> for license details for the 'C' code.

LazyData true

Repository CRAN

RoxygenNote 7.3.2

Encoding UTF-8

NeedsCompilation yes

Author Yanrong Song [aut, cre], Zijie Zhu [aut], David Kane [aut], Heidi Chen [aut], Yuanchu Dang [aut], Yang Lu [aut], Kanishka Malik [aut], Skylar Smith [aut], International Swaps and Derivatives Association [cph] (Copyright holder of the free CDS standard model code used in this package)

Maintainer Yanrong Song <yrsong129@gmail.com>

Date/Publication 2024-12-19 21:00:02 UTC

Contents

`add_conventions` *Return accounting conventions*

Description

`add_conventions` takes a data frame with a currency.var column and returns the same data frame with eight other columns of accounting conventions added to it.

Usage

```
add_conventions(x, currency.var = "currency")
```

Arguments

x	a data frame containing all necessary information
currency.var	a character indicating the name of currency column

Value

a data frame with eight more columns of accounting conventions: badDayConvention (a character indicating how non-business days are converted), mmDCC (the day count convention of the instruments), mmCalendars (any calendar adjustment for the CDS), fixedDCC (the day count convention of the fixed leg), floatDCC (the day count convention of the floating leg), fixedFreq (the frequency of the fixed rate of swap being paid), floatFreq (the frequency of the floating rate of swap being paid) and swapCalendars (any calendar adjustment for swap rate)

References

https://www.cdsmodel.com/cdsmodel/assets/cds-model/docs/c-code%20Key%20Functions-v1.pdf

Examples

```
x <- data.frame(date = c(as.Date("2014-05-06"), as.Date("2014-05-07")),
                currency = c("USD", "JPY"))
add_conventions(x)
```

---

add_dates *Return CDS dates.*

Description

add_dates takes a data frame which contains dates, tenor (or maturity) and currency and returns appropriate dates for pricing a CDS contract.

Usage

```
add_dates(
  x,
  date.var = "date",
  maturity.var = "maturity",
  tenor.var = "tenor",
  currency.var = "currency"
)
```

Arguments

x	a data frame, containing all necessary information
date.var	character, column name of date variable
maturity.var	character, column name of maturity variable
tenor.var	character, column name of tenor variable
currency.var	character, column name of currency variable

Value

a date frame containing all the input columns, as well as eight more columns: stepinDate (T+1), valueDate (T+3 business days), startDate (accrual begin date), endDate (maturity), backstopDate (T-60 day look back from which 'protection' is effective), firstcouponDate (the date on which the first coupon is paid), pencouponDate (second to last coupon date), and baseDate (the starting date for the IR curve)

References

https://www.cdsmodel.com/cdsmodel/assets/cds-model/docs/c-code%20Key%20Functions-v1.pdf https://www.cdsmodel.com/assets/cds-model/docs/Standard%20CDS%20Examples.pdf

Examples

```
x <- data.frame(date = c(as.Date("2014-05-06"), as.Date("2014-05-07")),
                tenor = rep(5, 2), currency = c("JPY", "USD"))
add_dates(x)
```

---

adj_next_bus_day *Adjust to next business day.*

Description

adj_next_bus_day gets the next business day following 5D bus day convention.

Usage

```
adj_next_bus_day(date)
```

Arguments

date	a Date type.

Value

Date adjusted to the following business day

build_rates *Build a data frame containing interest rates for CDS pricing*

Description

build_rates can create a data frame of interest rates from a start date to an end date, which are specified by the user. The interest rates will be later used in CDS pricing. build_rates also builds the rates.RData in the package. build_rates mainly consists download_markit and download_FRED. The two sources are Markit website and FRED website.

Usage

```
build_rates(start.date, end.date)
```

Arguments

start.date	is the start date of the data frame; it is the earliest CDS pricing date that the user concerns.
end.date	is the end date of the data frame; it is the lastest CDS pricing date that the user concerns.

Details

since Markit wesite has the interest rates back to th 1990s, download_markit is responsible for building up all the USD interest rate data frame; for EUR and JPY, markit can only get from 2005-01-05 to now. But the biggest advantage of using download_markit is that, since Markit website only lists the rate expiries that are actually used for CDS pricing, download_markit can get the exact type of expiries of rates needed to price CDS. Also, it has expiry over 1Y to 30Y. In contrast, FRED is only complementary to markit data. Since download_markit can get any rates for USD, we don't use FRED for USD. Then we want to get data for EUR and JPY before 2005-01-04 (the limit of markit). FRED has almost all data for any date, which is its biggest advantage. But its biggest disadvantage is that it doesn't know which expiry type is suitable for which time, since FRED website has expiry of all types below a year; also it doesn't have expiry over 1Y.So we hardcoded the dates below, to combine markit and FRED in the most suitable way.

Another note is that, the rates on both Markit website and FRED website have not been adjusted to the previous business day. In other words, the rates from both website is the exact rate on that day, rather than on the previous business day. But download_markit and download_FRED have adjusted the days already for the convenience of CDS pricing, so we don't have to worry here.

Value

a data frame that contains the CDS pricing date, the currency, the interest rate expiry and the interest rate.

References

https://www.spglobal.com/en https://fred.stlouisfed.org/docs/api/fred/

See Also

download_markit download_FRED rates

Examples

```
## Not run:
## Running this example will take more than two hours.

build_rates(start.date = as.Date("2004-01-01"),
            end.date = as.Date("2014-08-23"))
## End(Not run)
```

`call_ISDA` *call ISDA c function*

Description

`call_ISDA` call ISDA function

Usage

```
call_ISDA(x, name, rates.info)
```

Arguments

`x`	dataframe which contains relevant dates and convention info
`name`	character function name within which call_ISDA is called
`rates.info`	dataframe which contains relevant rates data

Value

a numeric value which is the difference between the new upfront and the old one

---

`CDS` *Build a* `CDS` *class object given the input about a CDS contract.*

Description

Build a `CDS` class object given the input about a CDS contract.

Usage

```
CDS(
  name = NULL,
  contract = "SNAC",
  RED = NULL,
  date = Sys.Date(),
  spread = NULL,
  maturity = NULL,
  tenor = NULL,
  coupon = 100,
  recovery = 0.4,
  currency = "USD",
  notional = 1e+07
)
```

Arguments

name	is the name of the reference entity. Optional.
contract	is the contract type, default SNAC
RED	alphanumeric code assigned to the reference entity. Optional.
date	is when the trade is executed, denoted as T. Default is Sys.Date. The date format should be in "YYYY-MM-DD".
spread	CDS par spread in bps.
maturity	date of the CDS contract.
tenor	of contract. By default is set as 5
coupon	quoted in bps. It specifies the payment amount from the protection buyer to the seller on a regular basis. The default is 100 bps.
recovery	in decimal. Default is 0.4.
currency	in which CDS is denominated.
notional	is the amount of the underlying asset on which the payments are based. Default is 10000000, i.e. 10MM.

Value

a CDS class object including the input informtion on the contract as well as the valuation results of the contract.

Examples

```
x <- CDS(date = as.Date("2014-05-07"), tenor = 5, spread = 50, coupon = 100)
```

CDS, CDS-class *CDS Class*

Description

Class definition for the CDS-Class

Slots

name is the name of the reference entity. Optional.

contract is the contract type, default SNAC

RED alphanumeric code assigned to the reference entity. Optional.

date is when the trade is executed, denoted as T. Default is Sys.Date.

spread CDS par spread in bps.

maturity date of the CDS contract.

tenor of contract in number of years - 5, 3

coupon quoted in bps. It specifies the payment amount from

recovery in decimal. Default is 0.4.

currency in which CDS is denominated.

principal is the dirty upfront less the accrual.

accrual is the accrued interest payment.

pd is the approximate the default probability at time t given the spread.

price is the price

upfront is quoted in the currency amount. Since a standard contract is traded with fixed coupons, upfront payment is introduced to reconcile the difference in contract value due to the difference between the fixed coupon and the conventional par spread. There are two types of upfront, dirty and clean. Dirty upfront, a.k.a. Cash Settlement Amount, refers to the market value of a CDS contract. Clean upfront is dirty upfront less any accrued interest payment, and is also called the Principal.

spread.DV01 measures the sensitivity of a CDS contract mark-to-market to a parallel shift in the term structure of the par spread.

IR.DV01 is the change in value of a CDS contract for a 1 bp parallel increase in the interest rate curve. IRDV01 is, typically, a much smaller dollar value than spreadDV01 because moves in overall interest rates have a much smaller effect on the value of a CDS contract than does a move in the CDS spread itself.

rec.risk.01 is the dollar value change in market value if the recovery rate used in the CDS valuation were increased by 1%.

check_inputs *Check whether inputs from the data frame are valid.*

Description

check_inputs checks whether a data frame's inputs are valid. It is a minimum set of checks. Things such as recovery var are not checked, because some functions don't need them as input.

Usage

```
check_inputs(
  x,
  date.var = "date",
  currency.var = "currency",
  maturity.var = "maturity",
  tenor.var = "tenor",
  spread.var = "spread",
  coupon.var = "coupon",
  notional.var = "notional",
  notional = 1e+06,
  recovery.var = "recovery",
  recovery = 0.4
)
```

Arguments

x	data frame, contains all the relevant columns.
date.var	character, column in x containing date variable.
currency.var	character, column in x containing currency.
maturity.var	character, column in x containing maturity date.
tenor.var	character, column in x containing tenors.
spread.var	character, column in x containing spread in basis points.
coupon.var	character, column in x containing coupon rates in basis points. It specifies the payment amount from the protection buyer to the seller on an annual basis.
notional.var	character, column in x containing the amount of the underlying asset on which the payments are based.
notional	numeric, the notional amount for all pricing if there isn't a notional.var
recovery.var	character, column in x containing recovery rates. ISDA model standard recovery rate asscumption is 0.4.
recovery	numeric, the recovery rate for all pricing if there isn't a recovery.var

Value

a data frame if not stopped by errors.

Examples

```
x <- data.frame(date = as.Date(c("2014-04-22", "2014-04-22")),
                currency = c("USD", "EUR"),
                tenor = c(5, 5),
                spread = c(120, 110),
                coupon = c(100, 100),
                recovery = c(0.4, 0.4),
                notional = c(1000000, 1000000))
x <- check_inputs(x)
```

`creditr` *The creditr package.*

Description

`creditr` package prices credit default swaps (CDS). It enables CDS class object which has slots as name, contract, RED, date, spread, maturity, teno, coupon, recovery, currency, notional, principal, accrual, pd, price, upfront, spread.DV01, IR.DV01 and rec.risk.01, with S4 methods like update, show and summary. It also supports data frame input and is able to provide convenient calculation of key CDS statistics through functions like `CS10`, `IR.DV01`, `rec_risk_01` and `spread_DV01`. Of other major functions, `spread_to_upfront` and `upfront_to_spread` are designed to compute one of spread and upfront given the other; `spread_to_pd` and `pd_to_spread`, similarly, can calculate one of spread and probability of default given the other; `add_dates` and `add_conventions` compute a series of dates information and accounting conventions related to CDS pricing. Finally, `get_rates` and `build_rates` facilitates direct fetching of relevant interest rates from online sources. Thanks to ISDA Standard Model's Open Source license, we are able to create this package for R users. You can find the Open Source licence of ISDA Standard Model at "https://www.cdsmodel.com/cdsmodel/cds-disclaimer.html"

Author(s)

Maintainer: Yanrong Song <yrsong129@gmail.com>

Authors:

- Zijie Zhu <zijie.miller.zhu@gmail.com>
- David Kane <dave.kane@gmail.com>
- Heidi Chen <s.heidi.chen@gmail.com>
- Yuanchu Dang <yuanchu.dang@gmail.com>
- Yang Lu <yang.lu2014@gmail.com>
- Kanishka Malik <kanishkamalik@gmail.com>
- Skylar Smith <sws2@williams.edu>

Other contributors:

- International Swaps and Derivatives Association (Copyright holder of the free CDS standard model code used in this package) [copyright holder]

See Also

Useful links:

- https://github.com/yanrong-stacy-song/creditr

`CS10` *Calculate CS10*

Description

`CS10` calculates the change in upfront value when the spread rises by 10

Usage

```
CS10(
  x,
  date.var = "date",
  currency.var = "currency",
  maturity.var = "maturity",
  tenor.var = "tenor",
  spread.var = "spread",
  coupon.var = "coupon",
  recovery.var = "recovery",
  notional.var = "notional",
  notional = 1e+07,
  recovery = 0.4
)
```

Arguments

`x`	data frame, contains all the relevant columns.
`date.var`	character, column in x containing date variable.
`currency.var`	character, column in x containing currency.
`maturity.var`	character, column in x containing maturity date.
`tenor.var`	character, column in x containing tenors.
`spread.var`	character, column in x containing spread in basis points.
`coupon.var`	character, column in x containing coupon rates in basis points. It specifies the payment amount from the protection buyer to the seller on an annual basis.
`recovery.var`	character, column in x containing recovery rates. ISDA model standard recovery rate asscumption is 0.4.
`notional.var`	character, column in x containing the amount of the underlying asset on which the payments are based.
`notional`	numeric, the notional amount for all pricing if there isn't a notional.var
`recovery`	numeric, the recovery rate for all pricing if there isn't a recovery.var

Value

a vector containing the change in upfront in units of currency.var when spread increase by 10

Examples

```
x <- data.frame(date = as.Date(c("2014-04-22", "2014-04-22")),
                currency = c("USD", "EUR"),
                tenor = c(5, 5),
                spread = c(120, 110),
                coupon = c(100, 100),
                recovery = c(0.4, 0.4),
                notional = c(10000000, 10000000),
                stringsAsFactors = FALSE)
CS10(x)
```

---

download_FRED *Get Rates from FRED*

Description

download_FRED returns the deposits and swap rates for the day input, along with the date conventions for that specific currency. The source is FRED.

Usage

```
download_FRED(
  start = as.Date("2004-01-01"),
  end = as.Date("2005-01-04"),
  currency = "JPY"
)
```

Arguments

start	is the start date of the data frame we want
end	is the end date of the data frame we want
currency	is the three-letter currency code. As of now, it works for USD, EUR, and JPY. The default is JPY.

Value

a data frame that contains the rates based on the ISDA pecifications

See Also

download_markit build_rates

Examples

```
## Not run:
download_FRED(start = as.Date("2003-12-31"), end = as.Date("2005-01-04"),
              currency = "JPY")

## End(Not run)
```

`download_markit` *Get rates from Markit*

Description

`download_markit` takes a data frame of dates and returns a data frame with the yields for different maturities.

Usage

```
download_markit(start, end, currency = "USD")
```

Arguments

`start`	date for gathering interest rates. Must be a Date type
`end`	date for gathering interest rates. Must be a Date type
`currency`	for which rates are being retrieved

Value

data frame containing the rates from every day from start to end dates. Note: the date in the output data frame does not refer to the rates of that day but to the date on which the CDS is being priced. So the corresponding rate is actually the rate of the previous day. Example: if the column reads 2014-04-22, the corresponding rates are actually for 2014-04-21.

See Also

`download_FRED` `build_rates`

Examples

```
## Not run:
download_markit(start = as.Date("2005-12-31"), end = as.Date("2006-01-04"),
                currency = "JPY")

## End(Not run)
```

get_rates *Get interest rates from rates.RData or the Markit website*

Description

get_rates returns the deposits and swap rates for the day input, along with the date conventions for that specific currency. The day input should be a weekday. If not, go to the most recent previous weekday.

Usage

```
get_rates(date, currency)
```

Arguments

date	Trade date. The rates for a trade date T are published on T-1 weekday. This date refers to the day on which we want the CDS to be priced, not the date for the interest rates as the interest rates will be used is the day before the trade date. Eg. If we are trying to find the rates used to price a CDS on 2014-04-22, it will return the rates of 2014-04-21
currency	the three-letter currency code. As of now, it works for USD, EUR, and JPY. The default is USD.

Value

a data frame that contains date (the CDS pricing date),

Examples

```
get_rates(as.Date("2014-05-07"), currency = "USD")
```

get_raw_markit *Get raw data from Markit website.*

Description

get_raw_markit downloads the rates zip file from Markit website, unzips and parses the XML

Usage

```
get_raw_markit(date, currency)
```

Arguments

date	Date type, is the CDS pricing date.
currency	numeric, is the currency that the CDS is traded in.

Value

a data frame that contains CDS pricing date, currency, interest rate expiry and interest rate. The data frame is created with data from Markit website

---

`implied_RR` *Calculates Implied Recovery Rate*

Description

`implied_RR` that calculates the recovery rate implied by the CDS spread and probability of default (pd) by using the ISDA model. This takes a data frame of inputs and returns a vector of the same length.

Usage

```
implied_RR(
  x,
  date.var = "date",
  tenor.var = "tenor",
  maturity.var = "maturity",
  spread.var = "spread",
  pd.var = "pd"
)
```

Arguments

`x`	data frame, contains all the relevant columns.
`date.var`	character, column in x containing date variable.
`tenor.var`	character, column in x containing tenors.
`maturity.var`	character, column in x containing maturity date.
`spread.var`	character, column in x containing spread in basis points.
`pd.var`	name of the column containing the probability of default rates.

Value

implied recovery rate in percentage based on the general approximation for a probability of default in the Bloomberg manual. The actual calculation uses a complicated bootstrapping process, so the results may be marginally different.

IR_DV01 *Calculate IR.DV01*

Description

IR_DV01 calculate the amount of change in upfront when there is a 1/1e4 increase in interest rate for a data frame of CDS contracts.

Usage

```
IR_DV01(
  x,
  date.var = "date",
  currency.var = "currency",
  maturity.var = "maturity",
  tenor.var = "tenor",
  spread.var = "spread",
  coupon.var = "coupon",
  recovery.var = "recovery",
  notional.var = "notional",
  notional = 1e+07,
  recovery = 0.4
)
```

Arguments

x	data frame, contains all the relevant columns.
date.var	character, column in x containing date variable.
currency.var	character, column in x containing currency.
maturity.var	character, column in x containing maturity date.
tenor.var	character, column in x containing tenors.
spread.var	character, column in x containing spread in basis points.
coupon.var	character, column in x containing coupon rates in basis points. It specifies the payment amount from the protection buyer to the seller on an annual basis.
recovery.var	character, column in x containing recovery rates. ISDA model standard recovery rate asscumption is 0.4.
notional.var	character, column in x containing the amount of the underlying asset on which the payments are based.
notional	numeric, the notional amount for all pricing if there isn't a notional.var
recovery	numeric, the recovery rate for all pricing if there isn't a recovery.var

Value

a vector containing the change in upfront when there is a 1/1e4 increase in interest rate, for each corresponding CDS contract.

Examples

```
x <- data.frame(date = c(as.Date("2014-04-22"), as.Date("2014-04-22")),
                currency = c("USD", "EUR"),
                tenor = c(5, 5),
                spread = c(120, 110),
                coupon = c(100, 100),
                recovery = c(0.4, 0.4),
                notional = c(10000000, 10000000),
                stringsAsFactors = FALSE)
IR_DV01(x)
```

`pd_to_spread` *Calculate spread with Default Probability*

Description

`pd_to_spread` to calculate spread using the probability of default, tenor and recovery rate.

Usage

```
pd_to_spread(
  x,
  recovery.var = "recovery",
  currency.var = "currency",
  tenor.var = "tenor",
  date.var = "date",
  pd.var = "pd"
)
```

Arguments

`x`	data frame, contains all the relevant columns.
`recovery.var`	character, column in x containing recovery rates. ISDA model standard recovery rate asscumption is 0.4.
`currency.var`	character, column in x containing currency.
`tenor.var`	character, column in x containing tenors.
`date.var`	character, column in x containing date variable.
`pd.var`	name of the column containing the probability of default in decimals.

Value

vector containing the spread values in basis points, calculated by inverting the formula for probability of default given in the Bloomberg Manual

See Also

`spread_to_pd`

PV01 *Calculate PV01*

Description

PV01 to calculate present value 01 or present value of a stream of 1bp payments

Usage

```
PV01(
  x,
  principal.var = "principal",
  spread.var = "spread",
  coupon.var = "coupon",
  notional.var = "notional"
)
```

Arguments

x	data frame, contains all the relevant columns.
principal.var	name of the column containing the principal or clean upfront values of the CDS
spread.var	character, column in x containing spread in basis points.
coupon.var	character, column in x containing coupon rates in basis points. It specifies the payment amount from the protection buyer to the seller on an annual basis.
notional.var	character, column in x containing the amount of the underlying asset on which the payments are based.

Value

Vector containing the PV01 values

rates *LIBOR rates from 2004-01-01 to 2015-08-03*

Description

This data frame is created by build_rates in creditr package to calculate the CDS pricing. It covers three currencies: USD, EUR, and JPY. The interest rates date from 2004-01-01 to 2014-08-23. Rates on holidays and weekends are available as well as business days.

Format

A data frame with 194378 observations on the following 4 variables.

- date = a date (Date object)
- currency = a character containing `USD EUR JPY`
- expiry = a character containing `1M 2M 3M 6M 9M 1Y 2Y 3Y 4Y 5Y 6Y 7Y 8Y 9Y 10Y 12Y 15Y 20Y 30Y`
- rate = a numeric vector. The LIBOR rate.

Details

The source of the interest rates in `rates.RData` is from https://www.spglobal.com/en and https://fred.stlouisfed.org/. When a user is calculating CDS using the `CDS` package, the package calls `get_rates` to get the needed interest rates; `get_rates` then calls the `rates.RData` for these interest rates. If a date is unavailable in `rates.RData`, then the package calls other functions to get the needed interest rates from the internet. The `rates.RData` is created and stored in the package for the users' convenience: getting interest rates from the Internet may fail due to the internet connection problem and may be very slow. Also, the user can build its own updated local `rates.RData` by using `build_rates`. For more explanation on the usage of `build_rates`, please see **See Also**.

Also, please notice that in the `rates.RData`, the `rate` is not the interest rate on the `date` listed in the same row as the `rate`. For example, in the first row, the `rate` is 0.001550; the `date` in that row is 2014-08-21. But this does not mean that on 2014-08-21, the interest rate is 0.001550; instead, 0.001550 is the interest rate on 2014-08-20. This regulation of using the interest rate on the previous business day of CDS trading date is set by ISDA Standard Model. The Model says that, if a trader buy a CDS on 2014-08-21, then when calculating the pricing of the CDS, she should use the interest rate of 2014-08-20, which is 0.001550. This may be confusing for people who are yet unfamiliar with CDS. We design `rates.RData` in this way because it is easier for other functions in the `CDS` package to use this data frame for calculation.

`rates.RData` covers holidays, weekend and business days. As is set by ISDA Standard Model and introduced above, we use the previous business day's interest rate for CDS pricing on a certain trading date. Therefore, if the user is buying a CDS on Saturday, she should use the interest rate of last Friday; if she is buying a CDS on Sunday or Monday, she should still use the interest rate of last Friday, because last Friday is the previous business day of the CDS trading date. When it comes to holidays, we still choose the previous business day for interest rate. For example, if a trader is buying a CDS on 2014-07-05, then she should use the interest rate of 2014-07-03, because 2014-07-04 is a national holiday and 2014-07-03 is the previous business day of trading date.

Also, please notice that in `rates.RData`, a currency's type of expiries generally stays the same along the time, but not always. For example, we check the expiry type of USD in `rates.RData`: for 2004-01-01, there are two types of expirty: 1M, 3Y; for 2006-01-01, there are five types of expiry: 1M, 2M, 3M, 6M, 1Y; for 2007-01-01, there are 18 types of expiry; for 2014-01-01, there are 19 types of expiry. Therefore, for a trader, she should always keep in mind to update her knowledge of the expiry types of her trading currency. For different currencies, the types of expiries are often different.

Finally, please notice that some of the data are missing for a certain expiry of a currency in a short time. For example, some dates in `rates.RData` do not have a `expiry` of 3Y for `USD`. This is not

likely to be caused by data error, since all these data are got from Markit and FRED. Users, however, should be aware of that some data seem "missing".

Source

https://www.spglobal.com/en https://fred.stlouisfed.org/

See Also

download_FRED download_markit build_rates

Examples

```
data(rates)

## for JPY rates:
rates[rates$currency == "JPY",]

## for rates on a specific date, of a specific currency:
rates[rates$currency == "USD" & rates$date == "2005-10-01",]
```

---

rec_risk_01 *Calculate Recovery Rate Changes*

Description

rec_risk_01 calculates the amount of change in upfront when there is a 1

Usage

```
rec_risk_01(
  x,
  date.var = "date",
  currency.var = "currency",
  maturity.var = "maturity",
  tenor.var = "tenor",
  spread.var = "spread",
  coupon.var = "coupon",
  recovery.var = "recovery",
  notional.var = "notional",
  recovery = 0.4,
  notional = 1e+07
)
```

Arguments

x	data frame, contains all the relevant columns.
date.var	character, column in x containing date variable.
currency.var	character, column in x containing currency.
maturity.var	character, column in x containing maturity date.
tenor.var	character, column in x containing tenors.
spread.var	character, column in x containing spread in basis points.
coupon.var	character, column in x containing coupon rates in basis points. It specifies the payment amount from the protection buyer to the seller on an annual basis.
recovery.var	character, column in x containing recovery rates. ISDA model standard recovery rate asscumption is 0.4.
notional.var	character, column in x containing the amount of the underlying asset on which the payments are based.
recovery	numeric, the recovery rate for all pricing if there isn't a recovery.var
notional	numeric, the notional amount for all pricing if there isn't a notional.var

Value

a vector containing the change in upfront when there is a 1 percent increase in recovery rate, for each corresponding CDS contract.

Examples

```
x <- data.frame(date = c(as.Date("2014-04-22"), as.Date("2014-04-22")),
                currency = c("USD", "EUR"),
                tenor = c(5, 5),
                spread = c(120, 110),
                coupon = c(100, 100),
                recovery = c(0.4, 0.4),
                notional = c(10000000, 10000000),
                stringsAsFactors = FALSE)
rec_risk_01(x)
```

---

separate_YMD *Separate Year/Month/Day*

Description

separate_YMD contains helper functions to separate an input date into year, month, and day.

Usage

```
separate_YMD(d)
```

Arguments

d	is an input date.

Value

an array contains year, month, date of the input date d.

---

show *Show Method*

Description

show shows a CDS class object.

Usage

```
## S4 method for signature 'CDS'
show(object)
```

Arguments

object	the input CDS object

---

spread_DV01 *Calculate Spread Change*

Description

spread_DV01 calculates the spread DV01 or change in upfront value when the spread rises by 1 basis point

Usage

```
spread_DV01(
  x,
  date.var = "date",
  currency.var = "currency",
  maturity.var = "maturity",
  tenor.var = "tenor",
  spread.var = "spread",
  coupon.var = "coupon",
  recovery.var = "recovery",
  notional.var = "notional",
  notional = 1e+07,
  recovery = 0.4
)
```

Arguments

x	data frame, contains all the relevant columns.
date.var	character, column in x containing date variable.
currency.var	character, column in x containing currency.
maturity.var	character, column in x containing maturity date.
tenor.var	character, column in x containing tenors.
spread.var	character, column in x containing spread in basis points.
coupon.var	character, column in x containing coupon rates in basis points. It specifies the payment amount from the protection buyer to the seller on an annual basis.
recovery.var	character, column in x containing recovery rates. ISDA model standard recovery rate asscumption is 0.4.
notional.var	character, column in x containing the amount of the underlying asset on which the payments are based.
notional	numeric, the notional amount for all pricing if there isn't a notional.var
recovery	numeric, the recovery rate for all pricing if there isn't a recovery.var

Value

a vector containing the change in upfront when there is a 1 basis point increase in spread, for each corresponding CDS contract.

Examples

```
x <- data.frame(date = c(as.Date("2014-04-22"), as.Date("2014-04-22")),
                currency = c("USD", "EUR"),
                tenor = c(5, 5),
                spread = c(120, 110),
                coupon = c(100, 100),
                recovery = c(0.4, 0.4),
                notional = c(10000000, 10000000),
                stringsAsFactors = FALSE)
spread_DV01(x)
```

---

spread_to_pd *Calcualte Default Probability with Spread*

Description

spread_to_pd approximates the default probability at time given the spread

Usage

```
spread_to_pd(
  x,
  recovery.var = "recovery",
  currency.var = "currency",
  tenor.var = "tenor",
  maturity.var = "maturity",
  date.var = "date",
  spread.var = "spread"
)
```

Arguments

x	data frame, contains all the relevant columns.
recovery.var	character, column in x containing recovery rates. ISDA model standard recovery rate asscumption is 0.4.
currency.var	character, column in x containing currency.
tenor.var	character, column in x containing tenors.
maturity.var	character, column in x containing maturity date.
date.var	character, column in x containing date variable.
spread.var	character, column in x containing spread in basis points.

Value

vector containing the probability of default, calculated by using the formula for probability of default given in the Bloomberg Manual

See Also

pd_to_spread

---

spread_to_upfront *Calculate Upfront Payments*

Description

spread_to_upfront takes a dataframe of variables on CDSs to return a vector of upfront values. Note that all CDS in the data frame must be denominated in the same currency.

Usage

```
spread_to_upfront(
  x,
  currency.var = "currency",
  notional = 1e+07,
  date.var = "date",
  spread.var = "spread",
  coupon.var = "coupon",
  tenor.var = "tenor",
  maturity.var = "maturity",
  recovery.var = "recovery",
  isPriceClean = FALSE
)
```

Arguments

x	data frame, contains all the relevant columns.
currency.var	character, column in x containing currency.
notional	is the amount of the underlying asset on which the payments are based. Default is 10000000, i.e. 10MM.
date.var	character, column in x containing date variable.
spread.var	character, column in x containing spread in basis points.
coupon.var	character, column in x containing coupon rates in basis points. It specifies the payment amount from the protection buyer to the seller on an annual basis.
tenor.var	character, column in x containing tenors.
maturity.var	character, column in x containing maturity date.
recovery.var	character, column in x containing recovery rates. ISDA model standard recovery rate asscumption is 0.4.
isPriceClean	refers to the type of upfront calculated. It is boolean. When TRUE, calculate principal only. When FALSE, calculate principal + accrual.

Value

vector of upfront values (with accrual) in the same order

---

summary *Summary Method*

Description

summary method displays only the essential info about the CDS class object.

Usage

```
## S4 method for signature 'CDS'
summary(object, ...)
```

Arguments

object	the input CDS object
...	additional arguments to pass in

---

upfront_to_spread *Calculate Spread with a Given Upfront*

Description

upfront_to_spread calculates conventional spread using the upfront or ptsUpfront values.

Usage

```
upfront_to_spread(
  x,
  currency.var = "currency",
  date.var = "date",
  coupon.var = "coupon",
  tenor.var = "tenor",
  maturity.var = "maturity",
  recovery.var = "recovery",
  upfront.var = "upfront",
  points.var = "ptsUpfront",
  isPriceClean = FALSE,
  notional = 1e+07,
  payAccruedAtStart = FALSE,
  payAccruedOnDefault = TRUE
)
```

Arguments

x	data frame, contains all the relevant columns.
currency.var	character, column in x containing currency.
date.var	character, column in x containing date variable.
coupon.var	character, column in x containing coupon rates in basis points. It specifies the payment amount from the protection buyer to the seller on an annual basis.
tenor.var	character, column in x containing tenors.
maturity.var	character, column in x containing maturity date.
recovery.var	character, column in x containing recovery rates. ISDA model standard recovery rate asscumption is 0.4.

upfront.var	is the character name of upfront column
points.var	character name of points Upfront column
isPriceClean	a boolean variable indicating whether the upfront is clean or dirty
notional	numeric variable indicating the notional value of the CDS contract
payAccruedAtStart	whether pay at start date the accrual amount
payAccruedOnDefault	whether pay in default scenario the accrual amount
recovery	numeric, the recovery rate for all pricing if there isn't a recovery.var

Value

a numeric indicating the spread.

Index