Package ‘BondValuation’

May 28, 2022

Title  Fixed Coupon Bond Valuation Allowing for Odd Coupon Periods and Various Day Count Conventions

Date  2022-05-28

Version  0.1.1

Description  Analysis of large datasets of fixed coupon bonds, allowing for irregular first and last coupon periods and various day count conventions. With this package you can compute the yield to maturity, the modified and Macaulay durations and the convexity of fixed-rate bonds. It provides the function AnnivDates, which can be used to evaluate the quality of the data and return time-invariant properties and temporal structure of a bond.

Depends  R (>= 2.15.1)

Imports  Rcpp, timeDate

LazyData  TRUE

License  GPL-3

RoxygenNote  7.2.0

LinkingTo  Rcpp

Encoding  UTF-8

NeedsCompilation  yes

Author  Djatschenko Wadim [aut, cre]

Maintainer  Djatschenko Wadim <wadim.djatschenko@gmx.de>

Repository  CRAN

Date/Publication  2022-05-28 18:30:02 UTC

R topics documented:

AccrInt .................................................. 2
AnnivDates ............................................. 9
BondVal.Price ......................................... 20
BondVal.Yield .......................................... 22
DP ......................................................... 25
List.DCC ................................................. 28
NonBusDays.Brazil .................................... 29
AccrInt (calculation of accrued interest)

Description

AccrInt returns the amount of interest accrued from some starting date up to some end date and the number of days of interest on the end date.

Usage

AccrInt(
  StartDate = as.Date(NA),
  EndDate = as.Date(NA),
  Coup = as.numeric(NA),
  DCC = as.numeric(NA),
  RV = as.numeric(NA),
  CpY = as.numeric(NA),
  Mat = as.Date(NA),
  YearNCP = as.Date(NA),
  EOM = as.numeric(NA),
  DateOrigin = as.Date("1970-01-01"),
  InputCheck = 1
)

Arguments

StartDate Calendar date on which interest accrual starts. Date class object with format "%Y-%m-%d". (required)
EndDate Calendar date up to which interest accrues. Date class object with format "%Y-%m-%d". (required)
Coup Nominal interest rate per year in percent. (required)
DCC The day count convention for interest accrual. (required)
RV The redemption value of the bond. Default: 100.
CpY Number of interest payments per year (non-negative integer; element of the set \{1,2,3,4,6,12\}). Default: 2.
Mat So-called "maturity date" i.e. date on which the redemption value and the final interest are paid. Date class object with format "%Y-%m-%d".
YearNCP Year figure of the next coupon payment date after EndDate.
EOM Boolean indicating whether the bond follows the End-of-Month rule.
DateOrigin Determines the starting point for the daycount in "Date" objects. Default: "1970-01-01".
InputCheck If 1, the input variables are checked for the correct format. Default: 1.
Assuming that there is no accrued interest on StartDate the function `AccrInt` computes the amount of interest accrued up to EndDate under the terms of the specified day count convention DCC. The function returns a list of two numerics `AccrInt`, and `DaysAccrued`. If `InputCheck = 1` the input variables are checked for the correct format. The core feature of this function is the proper handling of the day count conventions presented below. The type of the day count convention determines the amount of the accrued interest that has to be paid by the buyer in the secondary market if the settlement takes place between two coupon payment dates.

- Many different day count conventions are used in the market. Since there is no central authority that develops these conventions there is no standardized nomenclature. The tables below provide alternative names that often are used for the respective conventions. Type View(List.DCC) for a list of the day count methods currently implemented.

- Detailed descriptions of the conventions and their application may be found in Djatschenko (2018), and the other provided references.

### Day Count Conventions

#### Actual/Actual (ISDA)

<table>
<thead>
<tr>
<th>DCC</th>
<th>= 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>other names</td>
<td>Actual/Actual, Act/Act, Act/Act (ISDA)</td>
</tr>
<tr>
<td>references</td>
<td>ISDA (1998); ISDA (2006) section 4.16 (b)</td>
</tr>
</tbody>
</table>

#### Actual/Actual (ICMA)

<table>
<thead>
<tr>
<th>DCC</th>
<th>= 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>other names</td>
<td>Actual/Actual (ISMA), Act/Act (ISMA), Act/Act (ICMA), ISMA-99</td>
</tr>
<tr>
<td>references</td>
<td>ICMA Rule 251; ISDA (2006) section 4.16 (c); SWX (2003)</td>
</tr>
<tr>
<td>-----------</td>
<td>--------------------------------------------------</td>
</tr>
</tbody>
</table>

**Actual/Actual (AFB)**

<table>
<thead>
<tr>
<th>DCC</th>
<th>= 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>other names</td>
<td>AFB Method, Actual/Actual (Euro), Actual/Actual AFB FBF, ACT/365-366 (leap day)</td>
</tr>
<tr>
<td>references</td>
<td>ISDA (1998); EBF (2004)</td>
</tr>
</tbody>
</table>

**Actual/365L**

<table>
<thead>
<tr>
<th>DCC</th>
<th>= 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>other names</td>
<td>Act/365-366, ISMA-Year</td>
</tr>
<tr>
<td>references</td>
<td>ICMA Rule 251; SWX (2003)</td>
</tr>
</tbody>
</table>

**30/360**

<table>
<thead>
<tr>
<th>DCC</th>
<th>= 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>other names</td>
<td>360/360, Bond Basis, 30/360 ISDA</td>
</tr>
<tr>
<td>references</td>
<td>ISDA (2006) section 4.16 (f); MSRB (2017) Rule G-33</td>
</tr>
</tbody>
</table>

**30E/360**

<p>| DCC | = 6 |</p>
<table>
<thead>
<tr>
<th></th>
<th>other names</th>
<th>references</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Eurobond Basis, Special German (30S/360), ISMA-30/360</td>
<td>ICMA Rule 251; ISDA (2006) section 4.16 (g); SWX (2003)</td>
</tr>
<tr>
<td></td>
<td>DCC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>other names</td>
<td>none</td>
</tr>
<tr>
<td></td>
<td>360/360 (German Master); German (30/360)</td>
<td>ISDA (2006) section 4.16 (h)</td>
</tr>
<tr>
<td></td>
<td>DCC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>other names</td>
<td>30/360, US (30U/360), 30/360 (SIA)</td>
</tr>
<tr>
<td></td>
<td>references</td>
<td>EBF (2004); SWX (2003)</td>
</tr>
<tr>
<td></td>
<td>DCC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>other names</td>
<td>30/360, US (30U/360), 30/360 (SIA)</td>
</tr>
<tr>
<td></td>
<td>references</td>
<td>Mayle (1993); SWX (2003)</td>
</tr>
<tr>
<td></td>
<td>DCC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Actual/365 (Fixed)</td>
<td></td>
</tr>
</tbody>
</table>

AccrInt
<table>
<thead>
<tr>
<th>DCC</th>
<th>=</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>other names</td>
<td></td>
<td>Act/365 (Fixed), A/365 (Fixed), A/365F, English</td>
</tr>
<tr>
<td>references</td>
<td></td>
<td>ISDA (2006) section 4.16 (d); SWX (2003)</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Actual(NL)/365</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCC</td>
</tr>
<tr>
<td>other names</td>
</tr>
<tr>
<td>references</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Actual/360</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCC</td>
</tr>
<tr>
<td>other names</td>
</tr>
<tr>
<td>references</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>30/365</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCC</td>
</tr>
<tr>
<td>references</td>
</tr>
</tbody>
</table>

---

| Act/365 (Canadian Bond) |
**AccrInt**  
Accrued interest on EndDate, given the other characteristics.

**DaysAccrued**  
The number of days of interest from StartDate to EndDate.

**References**

Transactions - Supplement to the Derivatives Annex - Interest Rate Transactions.

2. Caputo Silva, Anderson, Lena Oliveira de Carvalho, and Octavio Ladeira de Medeiros, 2010, 
*Public Debt: The Brazilian Experience* (National Treasury Secretariat and World Bank, Brasilia, 
BR).

Fixed Coupon Bond Analysis Allowing for Irregular Periods and Various Day Count Conventions 


12. SWX Swiss Exchange and D. Christie, 2003, "Accrued Interest & Yield Calculations and Determination of Holiday Calendars".

Examples

```r
StartDate<-rep(as.Date("2011-08-31"),16)
EndDate<-rep(as.Date("2012-02-29"),16)
Coup<-rep(5.25,16)
DCC<-seq(1,16)
RV<-rep(100000,16)
CpY<-rep(2,16)
Mat<-rep(as.Date("2021-08-31"),16)
YearNCP<-rep(2012,16)
EOM<-rep(1,16)

DCC_Comparison<-data.frame(StartDate,EndDate,Coup,DCC,RV,CpY,Mat,YearNCP,EOM)

AccrIntOutput<-apply(DCC_Comparison[,c("StartDate","EndDate","Coup","DCC","RV","CpY","Mat","YearNCP","EOM")],1,function(y) AccrInt(y[1],y[2],y[3],y[4],y[5],y[6],y[7],y[8],y[9]))

# warnings are due to apply's conversion of the variables' classes in DCC_Comparison to class "character"
Accrued_Interest<-do.call(rbind,lapply(AccrIntOutput, function(x) x[[1]]))
Days_Accrued<-do.call(rbind,lapply(AccrIntOutput, function(x) x[[2]]))
DCC_Comparison<-cbind(DCC_Comparison,Accrued_Interest,Days_Accrued)
DCC_Comparison
```
AnnivDates

AnnivDates returns a bond’s time-invariant characteristics and temporal structure as a list of three or four named data frames.

Usage

AnnivDates(
  Em = as.Date(NA),
  Mat = as.Date(NA),
  CpY = as.numeric(NA),
  FIPD = as.Date(NA),
  LIPD = as.Date(NA),
  FIAD = as.Date(NA),
  RV = as.numeric(NA),
  Coup = as.numeric(NA),
  DCC = as.numeric(NA),
  EOM = as.numeric(NA),
  DateOrigin = as.Date("1970-01-01"),
  InputCheck = 1,
  FindEOM = FALSE,
  RegCF.equal = 0
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Em</td>
<td>The bond’s issue date. (required)</td>
</tr>
<tr>
<td>Mat</td>
<td>Maturity date, i.e. date on which the redemption value and the final interest are paid. (required)</td>
</tr>
<tr>
<td>CpY</td>
<td>Number of interest payments per year (non-negative integer; element of the set {0,1,2,3,4,6,12}. Default: 2.</td>
</tr>
<tr>
<td>FIPD</td>
<td>First interest payment date after Em.</td>
</tr>
<tr>
<td>LIPD</td>
<td>Last interest payment date prior to Mat.</td>
</tr>
<tr>
<td>FIAD</td>
<td>Date on which the interest accrual starts (so-called &quot;dated date&quot;).</td>
</tr>
<tr>
<td>RV</td>
<td>The redemption value of the bond. Default: 100.</td>
</tr>
<tr>
<td>Coup</td>
<td>Nominal interest rate per year in percent. Default: NA.</td>
</tr>
<tr>
<td>DCC</td>
<td>The day count convention the bond follows. Default: NA. For a list of day count conventions currently implemented type View(List.DCC).</td>
</tr>
<tr>
<td>EOM</td>
<td>Boolean indicating whether the bond follows the End-of-Month rule. Default: NA.</td>
</tr>
</tbody>
</table>
DateOrigin  Determines the starting point for the daycount in "Date" objects. Default: "1970-01-01".
InputCheck  If 1, the input variables are checked for the correct format. Default: 1.
FindEOM  If TRUE, EOM is overridden by the value inferred from the data. Default: FALSE.
RegCF.equal  If 0, the amounts of regular cash flows are calculated according to the stipulated DCC. Any other value forces all regular cash flows to be equal sized. Default: 0.

Details  

AnnivDates  generates a list of the three data frames Warnings, Traits and DateVectors. If the variable Coup is passed to the function, the output contains additionally the data frame PaySched.  AnnivDates  is meant to analyze large data frames. Therefore some features are implemented to evaluate the quality of the data. The output of these features is stored in the data frame Warnings. Please see section Value for a detailed description of the tests run and the meaning of the variables in Warnings. The data frame Traits contains all time-invariant bond characteristics that were either provided by the user or calculated by the function. The data frame DateVectors contains three vectors of Date-Objects named RealDates, CoupDates and AnnivDates and three vectors of numerics named RD_indexes, CD_indexes and AD_indexes. These vectors are used in the other functions of this package according to the methodology presented in Djatschenko (2018). The data frame PaySched matches CoupDates to the actual amount of interest that the bond pays on the respective interest payment date. Section Value provides further information on the output of the function AnnivDates. Below information on the proper input format is provided. Subsequently follows information on the operating principle of the function AnnivDates and on the assumptions that are met to estimate the points in time needed to evaluate a bond.

• The dates Em, Mat, FIPD, LIPD and FIAD can be provided as
  1. "Date" with format "%Y-%m-%d", or
  2. "numeric" with the appropriate DateOrigin, or
  3. number of class "character" with the appropriate DateOrigin, or
  4. string of class "character" in the format "yyyy-mm-dd".

CpY, RV and Coup can be provided either as class "numeric" or as a number of class "character".

• The provided issue date (Em) is instantly substituted by the first interest accrual date (FIAD) if FIAD is available and different from Em.

• Before the determination of the bond’s date characteristics begins, the code evaluates the provided calendar dates for plausibility. In this process implausible dates are dropped. The sort of corresponding implausibility is identified and stored in a warning flag. (See section Value for details.)

• The remaining valid calendar dates are used to gauge whether the bond follows the End-of-Month-Rule. The resulting parameter est_EOM can take on the following values:

  **Case 1:** FIPD and LIPD are both NA

  | est_EOM = 1 | , if Mat is the last day of a month. |
  | est_EOM = 0 | , else. |
  | =========== | ============== |
Case 2: FIPD is NA and LIPD is a valid calendar date

| est_EOM = 1 | , if LIPD is the last day of a month. |
| est_EOM = 0 | , else. |

Case 3: FIPD is a valid calendar date and LIPD is NA

| est_EOM = 1 | , if FIPD is the last day of a month. |
| est_EOM = 0 | , else. |

Case 4: FIPD and LIPD are valid calendar dates

| est_EOM = 1 | , if LIPD is the last day of a month. |
| est_EOM = 0 | , else. |

- If EOM is initially missing or NA or not element of \( \{0, 1\} \), EOM is set est_EOM with a warning.
- If the initially provided value of EOM deviates from est_EOM, the following two cases apply:

  Case 1: If EOM = 0 and est_EOM = 1:
            EOM is not overridden and remains EOM = 0

  Case 2: If EOM = 1 and est_EOM = 0:
            EOM is overridden and set EOM = 0 with a warning.
            Keeping EOM = 1 in this case would conflict with
            the provided Mat, FIPD or LIPD.

Note: Set the option FindEOM=TRUE to always use est_EOM found by the code.

- If FIPD and LIPD are both available, the lengths of the first and final coupon periods are
determinate and can be "regular", "long" or "short". To find the interest payment dates between
FIPD and LIPD the following assumptions are met:

1. The interest payment dates between FIPD and LIPD are evenly distributed.
2. The value of EOM determines the location of all interest payment dates.

If assumption 1 is violated, the exact locations of the interest payment dates between FIPD and LIPD are ambiguous. The assumption is violated particularly, if

1. FIPD and LIPD are in the same month of the same year but not on the same day, or
2. the month difference between FIPD and LIPD is not a multiple of the number of months implied by CpY, or
3. FIPD and LIPD are not both last day in month, their day figures differ and the day figure difference between FIPD and LIPD is not due to different month lengths.

In each of the three cases, FIPD and LIPD are dropped with the flag IPD_CpY_Corrupt = 1.

• If neither FIPD nor LIPD are available the code evaluates the bond based only upon the required variables Em and Mat (and CpY, which is 2 by default). Since FIPD is not given, it is impossible to distinguish between a "short" and "long" odd first coupon period, without an assumption on the number of interest payment dates. Consequently the first coupon period is assumed to be either "regular" or "short". The locations of FIPD and LIPD are estimated under the following assumptions:

1. The final coupon period is "regular".
2. The interest payment dates between the estimated FIPD and Mat are evenly distributed.
3. The value of EOM determines the location of all interest payment dates.

• If LIPD is available but FIPD is not, the length of the final coupon payment period is determined by LIPD and Mat and can be "regular", "long" or "short". The locations of the interest payment dates are estimated under the following assumptions:

1. The first coupon period is either "regular" or "short".
2. The interest payment dates between the estimated FIPD and LIPD are evenly distributed.
3. The value of EOM determines the location of all interest payment dates.

• If FIPD is available but LIPD is not, the length of the first coupon payment period is determined by Em and FIPD and can be "regular", "long" or "short". The locations of the interest payment dates are estimated under the following assumptions:

1. The final coupon period is either "regular" or "short".
2. The interest payment dates between FIPD and the estimated LIPD are evenly distributed.
3. The value of EOM determines the location of all interest payment dates.

Value

All dates are returned irrespective of whether they are on a business day or not.
**DateVectors (data frame)** -

*RealDates* A vector of Date class objects with format "%Y-%m-%d" in ascending order, that contains the issue date, all actual coupon payment dates and the maturity date.

*RD_indexes* A vector of numerics capturing the temporal structure of the bond.

*CoupDates* A vector of Date class objects with format "%Y-%m-%d" in ascending order, that contains all actual coupon payment dates and the maturity date.

*CD_indexes* A vector of numerics capturing the temporal structure of the bond.

*AnnivDates* A vector of Date class objects with format "%Y-%m-%d" in ascending order, that contains all theoretical coupon anniversary dates. The first value of AnnivDates is the anniversary date immediately preceding the issue date, if the bond has an irregular first coupon period; otherwise it is the issue date. The final value of AnnivDates is the anniversary date immediately succeeding the maturity date, if the bond has an irregular final coupon period; otherwise it is the maturity date.

*AD_indexes* A vector of numerics capturing the temporal structure of the bond.

---

**PaySched (data frame)** -

*CoupDates* A vector of Date class objects with format "%Y-%m-%d" in ascending order, that contains all actual coupon payment dates and the maturity date.

*CoupPayments* A vector of class "numeric" objects, that contains the actual amounts of interest that the bond pays on the respective coupon payment dates. The unit of these payments is the same as that of RV that was passed to the function. RV is not included in the final interest payment.

**NOTE:** PaySched is created only if the variable Coup is provided.

---

**Traits (data frame)** -

*DateOrigin* The starting point for the daycount in "Date" objects.

*CpY* Number of interest payments per year.

*FIAD* Date on which the interest accrual starts (so-called "dated date").

*Em* The bond’s issue date that was used for calculations.

*Em_Orig* The bond’s issue date that was entered.

*FIPD* The first interest payment date after Em that was used for calculations. If the entered FIPD was dropped during the calculation process, the value is NA.

*FIPD_Orig* The first interest payment date after Em that was entered.

*est_FIPD* The estimated first interest payment date after Em, NA, if a valid FIPD was entered.

*LIPD* The last interest payment date prior to Mat that was used for calculations. If the entered LIPD was dropped during the calculation process, the value is NA.

*LIPD_Orig* The last interest payment date prior to Mat that was entered.

*est_LIPD* The estimated last interest payment date prior to Mat. NA, if a valid LIPD was entered.

*Mat* The maturity date that was entered.

*Refer* Reference date that determines the day figures of all AnnivDates.

*FCPType* A character string indicating the type of the first coupon period. Values: "long", "regular", "short".

*FCPLength* Length of the first coupon period as a fraction of a regular coupon period.
**LCPType**  A character string indicating the type of the last coupon period. Values: "long", "regular", "short".

**LCPLength**  Length of the final coupon period as a fraction of a regular coupon period.

**Par**  The redemption value of the bond.

**CouponInPercent.p.a**  Nominal interest rate per year in percent.

**DayCountConvention**  The day count convention the bond follows.

**EOM_Orig**  The value of EOM that was entered.

**est_EOM**  The estimated value of EOM.

**EOM_used**  The value of EOM that was used in the calculations.

**Warnings** (data frame) - A set of flags that indicate the occurrence of warnings during the execution. Below they are listed according to the hierarchical structure within the function AnnivDates.

---

**Em_FIAD_differ**

1

, if the provided issue date (Em) was substituted by the first interest accrual date (FIAD).

This happens, if FIAD is available and different from Em.

---

**Note:** No warning is displayed.

---

**EmMatMissing**

1

, if either issue date (Em) or maturity date (Mat) or both are missing or NA.

---

**Output:** RealDates = NA, CoupDates = NA,
AnnivDates = NA, FCPType = NA, LCPType = NA.

---

**CpYOverride**

1

, if number of interest periods per year (CpY) is missing or NA, or if the provided CpY is not element of \{0,1,2,3,4,6,12\}.
### AnnivDates

Note: CpY is set 2, and the execution continues.

Output: as if CpY = 2 was provided initially.

<table>
<thead>
<tr>
<th>RV_set100percent</th>
<th>0</th>
<th>else.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>===</td>
<td>---------------------------------------------</td>
</tr>
</tbody>
</table>

Note: RV is set 100, and the execution continues.

Output: as if RV = 100 was provided initially.

<table>
<thead>
<tr>
<th>NegLifeFlag</th>
<th>0</th>
<th>else.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>===</td>
<td>---------------------------------------------</td>
</tr>
</tbody>
</table>

Output: RealDates = NA, CoupDates = NA, AnnivDates = NA, FCPType = NA, LCPType = NA.

<table>
<thead>
<tr>
<th>ZeroFlag</th>
<th>0</th>
<th>else.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>===</td>
<td>---------------------------------------------</td>
</tr>
</tbody>
</table>

Output: RealDates = (Em, Mat), CoupDates = Mat, AnnivDates = (Em, Mat), FCPType = NA, LCPType = NA.
\[ Em_{\text{Mat}}_{\text{SameMY}} = 1 \implies \text{if the issue date (} Em \text{) and the maturity date (} Mat \text{) are in the same month of the same year but not on the same day, while } CpY \text{ is an element of } \{1,2,3,4,6,12\}. \]

Output: \( RealDates = (Em,Mat), CoupDates = Mat, FCPType = \text{short, LCPType = short.} \)

\[ 0 \implies \text{else.} \]

\[ ChronErrorFlag = 1 \implies \text{if the provided dates are in a wrong chronological order.} \]

\[ 0 \implies \text{else.} \]

\[ FIPD_{\text{LIPD}}_{\text{equal}} = 1 \implies \text{if } Em < FIPD = LIPD < Mat. \]

Output: \( AnnivDates \text{ contains } FIPD \text{ and has at least 3 elements. } RealDates = (Em,FIPD,Mat), CoupDates = (FIPD,Mat). FCPType \text{ and } LCPType \text{ can be "short", "regular" or "long".} \)

\[ 0 \implies \text{else.} \]

\[ IPD_{\text{CpY}}_{\text{Corrupt}} = 1 \implies \text{if the provided first interest payment date (} FIPD \text{) and last interest payment date (} LIPD \text{) are inconsistent with the provided number of interest payments per year (} CpY \text{).} \]
AnnivDates

Note:
Inconsistency occurs if
1. FIPD and LIPD are in the same month of the same year but not on the same day, or
2. the number of months between FIPD and LIPD is not a multiple of the number of months implied by CpY, or
3. FIPD and LIPD are not both last day in month, their day figures differ and the day figure difference between FIPD and LIPD is not due to different month lengths.

In each of the three cases keeping the provided values of FIPD and LIPD would violate the assumption, that the anniversary dates between FIPD and LIPD are evenly distributed.

FIPD and LIPD are set as.Date(NA) and the execution continues.

Output:
as if FIPD and LIPD were not provided initially.

\[ EOM\_Deviation = 1 \]

, if the provided value of EOM deviates from the value that is inferred from the provided calendar dates.

Note:
The program analyses the valid values of Em, Mat, FIPD and LIPD to determine the appropriate value of EOM.

If the initially provided value of EOM deviates from the value determined by the program, there might be an inconsistency in the provided data.

\[ EOM\_Override = 1 \]

, if the provided value of EOM is overridden by a value that
is inferred from the provided calendar dates.

Note:
This happens automatically if EOM is initially missing or NA or not element of \{0, 1\} and if the provided value of EOM conflicts with the provided values of FIPD, LIPD or Mat, e.g. if est\_EOM = 0 but EOM = 1.
If EOM\_Deviation = 1 and the option FindEOM is set TRUE, the initially provided value of EOM is also overridden by the value that is inferred from the provided calendar dates if est\_EOM = 1 but EOM = 0.

Output:
as if the value of EOM that is found by the program was provided initially.

\[
\begin{array}{ccc}
0 & \text{else.} & \text{else.} \\
\end{array}
\]

\[
\begin{array}{c}
DCC\_Override = 1 \\
\end{array}
\]
if DCC is missing or NA or not element of c(1:16).

Note:
If the program cannot process the provided day count identifier DCC, it overrides it with DCC = 2.

Output:
as if DCC = 2 was provided initially.

\[
\begin{array}{ccc}
0 & \text{else.} & \text{else.} \\
\end{array}
\]

NoCoups = 1, if there are no coupon payments between the provided issue date (Em) and the maturity date (Mat), but the provided (CpY) is not zero.

Output:
RealDates = (Em, Mat), CoupDates = (Mat), AnnivDates contains Mat and has either 2 or 3 elements, FCPType = LCPType and can be "short", "regular" or "long".
References


Examples

data(SomeBonds2016)

# Applying the function AnnivDates to the data frame SomeBonds2016.

system.time(
  FullAnalysis<-apply(SomeBonds2016[,c('Issue.Date','Mat.Date','CpY.Input','FIPD.Input','LIPD.Input','FIAD.Input','RV.Input','Coup.Input','DCC.Input','EOM.Input')],1,function(y)
    AnnivDates(y[1],y[2],y[3],y[4],y[5],y[6],y[7],y[8],y[9],y[10],RegCF.equal=1)),
  gcFirst = TRUE)
# warnings are due to apply's conversion of the variables' classes in # SomeBonds2016 to class "character"

# The output stored in FullAnalysis is a nested list.
# Lets look at what is stored in FullAnalysis for a random bond:
randombond<-sample(c(1:nrow(SomeBonds2016)),1)
FullAnalysis[[randombond]]

# Extracting the data frame Warnings:
AllWarnings<-do.call(rbind,lapply(FullAnalysis, \[\[, 1))
summary(AllWarnings)
# binding the Warnings to the bonds
BondsWithWarnings<-cbind(SomeBonds2016,AllWarnings)

# Extracting the data frame Traits:
AllTraits<-do.call(rbind,lapply(FullAnalysis, \[\[, 2))
summary(AllTraits)
# binding the Traits to the bonds
BondsWithTraits<-cbind(SomeBonds2016,AllTraits)

# Extracting the data frame AnnivDates:
AnnivDates<-lapply(lapply(FullAnalysis, \[\[, 3), \[\[, 5)
AnnivDates<-lapply(AnnivDates, \[length<-\, max(lengths(AnnivDates)))
AnnivDates<-as.data.frame(do.call(rbind, AnnivDates))
AnnivDates<-as.data.frame(lapply(AnnivDates, as.Date, as.Date(AllTraits$DateOrigin[1])))
# binding the AnnivDates to the bonds
BondsWithAnnivDates<-cbind(SomeBonds2016,AnnivDates)

# Extracting the data frames PaySched for each bond and creating a panel:
CoupSched<-lapply(FullAnalysis, \[\[, 4)
BondVal.Price

Description

**BondVal.Price** computes a bond’s clean price given its yield.

Usage

```r
BondVal.Price(
  YtM = as.numeric(NA),
  SETT = as.Date(NA),
  Em = as.Date(NA),
  Mat = as.Date(NA),
  CpY = as.numeric(NA),
  FIPD = as.Date(NA),
  LIPD = as.Date(NA),
  FIAD = as.Date(NA),
  RV = as.numeric(NA),
  Coup = as.numeric(NA),
  DCC = as.numeric(NA),
  EOM = as.numeric(NA),
  DateOrigin = as.Date("1970-01-01"),
  InputCheck = 1,
  FindEOM = FALSE,
  RegCF.equal = 0,
  SimpleLastPeriod = TRUE,
  Calc.Method = 1,
  AnnivDatesOutput = as.list(NA)
)
```

Arguments

- **YtM**: The bond’s yield to maturity p.a. on SETT. (required)
- **SETT**: The settlement date. Date class object with format "%Y-%m-%d". (required)
- **Em**: The bond’s issue date. Date class object with format "%Y-%m-%d". (required)
- **Mat**: So-called "maturity date" i.e. date on which the redemption value and the final interest are paid. Date class object with format "%Y-%m-%d". (required)
- **CpY**: Number of interest payments per year (non-negative integer; element of the set \{0,1,2,3,4,6,12\}). Default: 2.
**BondVal.Price**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIPD</td>
<td>First interest payment date after Em. Date class object with format &quot;%Y-%m-%d&quot;. Default: NA.</td>
</tr>
<tr>
<td>LIPD</td>
<td>Last interest payment date before Mat. Date class object with format &quot;%Y-%m-%d&quot;. Default: NA.</td>
</tr>
<tr>
<td>FIAD</td>
<td>Date on which the interest accrual starts (so-called &quot;dated date&quot;). Date class object with format &quot;%Y-%m-%d&quot;. Default: NA.</td>
</tr>
<tr>
<td>RV</td>
<td>The redemption value of the bond. Default: 100.</td>
</tr>
<tr>
<td>Coup</td>
<td>Nominal interest rate per year in percent. Default: NA.</td>
</tr>
<tr>
<td>DCC</td>
<td>The day count convention the bond follows. Default: NA. For a list of day count conventions currently implemented type View(List.DCC).</td>
</tr>
<tr>
<td>EOM</td>
<td>Boolean indicating whether the bond follows the End-of-Month rule. Default: NA.</td>
</tr>
<tr>
<td>DateOrigin</td>
<td>Determines the starting point for the daycount in &quot;Date&quot; objects. Default: &quot;1970-01-01&quot;.</td>
</tr>
<tr>
<td>InputCheck</td>
<td>If 1, the input variables are checked for the correct format. Default: 1.</td>
</tr>
<tr>
<td>FindEOM</td>
<td>If TRUE, EOM is overridden by the value inferred from the data. Default: FALSE.</td>
</tr>
<tr>
<td>RegCF.equal</td>
<td>If 0, the amounts of regular cash flows are calculated according to the stipulated DCC. Any other value forces all regular cash flows to be equal sized. Default: 0.</td>
</tr>
<tr>
<td>SimpleLastPeriod</td>
<td>Specifies the interest calculation method in the final coupon period. Default: TRUE.</td>
</tr>
<tr>
<td>Calc.Method</td>
<td>If 1, discount powers are computed with the same DCC as accrued interest. If 0, discount powers are computed with DCC=2. Default: 1.</td>
</tr>
<tr>
<td>AnnivDatesOutput</td>
<td>A list containing the output of the function AnnivDates. Default: NA.</td>
</tr>
</tbody>
</table>

**Details**

The function **BondVal.Price** uses the function **AnnivDates** to analyze the bond and computes the clean price, the accrued interest, the dirty price and the sensitivity measures modified duration (ModDUR), MacAulay duration (MacDUR) and convexity according to the methodology presented in Djatschenko (2018).

**Value**

- **CP** The bond’s clean price.
- **AccrInt** The amount of accrued interest.
- **DP** The bond’s dirty price.
- **ytm.p.a.** Annualized yield to maturity.
- **ModDUR.inYears** Modified duration in years.
- **MacDUR.inYears** MacAulay duration in years.
- **Conv.inYears** Convexity in years.
- **ModDUR.inPeriods** Modified duration in periods.
MacDUR.inPeriods  Macaulay duration in periods.
Conv.inPeriods  Convexity in periods.
tau  Relative Position of the settlement date in regular periods.

References


Examples

data(PanelSomeBonds2016)
randombond<-sample(c(1:length(which(!(duplicated(PanelSomeBonds2016$ID.No))))),1)
df.randombond<-PanelSomeBonds2016[which(PanelSomeBonds2016$ID.No==randombond),]

PreAnalysis.randombond<-suppressWarnings(AnnivDates(
    unlist(df.randombond[,c('Issue.Date','Mat.Date','CpY.Input','FIPD.Input','LIPD.Input','FIAD.Input','RV.Input','Coup.Input','DCC.Input','EOM.Input'),use.names=FALSE])))

system.time(
    for (i in c(1:nrow(df.randombond))) {
        BondVal.Price.Output<-suppressWarnings(BondVal.Price(
            unlist(df.randombond)[i,c('YtM.Input','TradeDate','Issue.Date','Mat.Date','CpY.Input','FIPD.Input','LIPD.Input','FIAD.Input','RV.Input','Coup.Input','DCC.Input','EOM.Input'),use.names=FALSE), 
            AnnivDatesOutput=PreAnalysis.randombond))
    }
)
plot(seq(1,nrow(df.randombond),by=1),df.randombond$CP.Out,"l")

---

**BondVal.Yield**

_BondVal.Yield (calculation of YiM, AccrInt, DP, ModDUR, MacDUR and Conv)_

**Description**

_BondVal.Yield_ returns a bond’s yield to maturity given its clean price.
Usage

BondVal.Yield(
    CP = as.numeric(NA),
    SETT = as.Date(NA),
    Em = as.Date(NA),
    Mat = as.Date(NA),
    CpY = as.numeric(NA),
    FIPD = as.Date(NA),
    LIPD = as.Date(NA),
    FIAD = as.Date(NA),
    RV = as.numeric(NA),
    Coup = as.numeric(NA),
    DCC = as.numeric(NA),
    EOM = as.numeric(NA),
    DateOrigin = as.Date("1970-01-01"),
    InputCheck = 1,
    FindEOM = FALSE,
    RegCF.equal = 0,
    SimpleLastPeriod = TRUE,
    Precision = .Machine$double.eps^0.75,
    Calc.Method = 1,
    AnnivDatesOutput = as.list(NA)
)

Arguments

CP
The bond’s clean price on SETT. (required)

SETT
The settlement date. Date class object with format "%Y-%m-%d". (required)

Em
The bond’s issue date. Date class object with format "%Y-%m-%d". (required)

Mat
So-called "maturity date" i.e. date on which the redemption value and the final
interest are paid. Date class object with format "%Y-%m-%d". (required)

CpY
Number of interest payments per year (non-negative integer; element of the set
\{0,1,2,3,4,6,12\}. Default: 2.

FIPD
First interest payment date after Em. Date class object with format "%Y-%m-%d". Default: NA.

LIPD
Last interest payment date before Mat. Date class object with format "%Y-%m-%d". Default: NA.

FIAD
Date on which the interest accrual starts (so-called "dated date"). Date class
object with format "%Y-%m-%d". Default: NA.

RV
The redemption value of the bond. Default: 100.

Coup
Nominal interest rate per year in percent. Default: NA.

DCC
The day count convention the bond follows. Default: NA. For a list of day count
conventions currently implemented type View(List.DCC).

EOM
Boolean indicating whether the bond follows the End-of-Month rule. Default: NA.
**DateOrigin**
Determines the starting point for the daycount in "Date" objects. Default: "1970-01-01".

**InputCheck**
If 1, the input variables are checked for the correct format. Default: 1.

**FindEOM**
If TRUE, EOM is overridden by the value inferred from the data. Default: FALSE.

**RegCF.equal**
If 0, the amounts of regular cash flows are calculated according to the stipulated DCC. Any other value forces all regular cash flows to be equal sized. Default: 0.

**SimpleLastPeriod**
Specifies the interest calculation method in the final coupon period. Default: TRUE.

**Precision**
desired precision in YtM-calculation. Default: .Machine$double.eps^0.75.

**Calc.Method**
If 1, discount powers are computed with the same DCC as accrued interest. If 0, discount powers are computed with DCC=2. Default: 1.

**AnnivDatesOutput**
A list containing the output of the function AnnivDates. Default: NA.

**Details**

**BondVal.Yield** uses the function **AnnivDates** to analyze the bond and computes the yield to maturity, the accrued interest, the dirty price and the sensitivity measures modified duration (Mod-DUR), MacAulay duration (MacDUR) and convexity according to the methodology presented in Djatschenko (2018). The yield to maturity is determined numerically using the Newton-Raphson method.

**Value**

- **CP** The bond’s clean price.
- **AccrInt** The amount of accrued interest.
- **DP** The bond’s dirty price.
- **ytm.p.a.** Annualized yield to maturity.
- **ModDUR.inYears** Modified duration in years.
- **MacDUR.inYears** MacAulay duration in years.
- **Conv.inYears** Convexity in years.
- **ModDUR.inPeriods** Modified duration in periods.
- **MacDUR.inPeriods** MacAulay duration in periods.
- **Conv.inPeriods** Convexity in periods.
- **tau** Relative Position of the settlement date in regular periods.

**References**

Examples

data(PanelSomeBonds2016)
randombond<-sample(c(1:length(which(!(duplicated(PanelSomeBonds2016$ID.No))))),1)
df.randombond<-PanelSomeBonds2016[which(PanelSomeBonds2016$ID.No==randombond),]

PreAnalysis.randombond<-suppressWarnings(AnnivDates(
  unlist(df.randombond[1,c('Issue.Date','Mat.Date','CpY.Input','FIPD.Input','LIPD.Input','FIAD.Input','RV.Input','Coup.Input','DCC.Input','EOM.Input')],
  use.names=FALSE)))

system.time(
  for (i in 1:nrow(df.randombond)) {
    BondVal.Yield.Output<-suppressWarnings(BondVal.Yield(
      unlist(df.randombond[i,c('CP.Input','TradeDate','Issue.Date','Mat.Date','CpY.Input','FIPD.Input','LIPD.Input','FIAD.Input','RV.Input','Coup.Input','DCC.Input','EOM.Input')],use.names=FALSE),
      AnnivDatesOutput=PreAnalysis.randombond))
  }
)
plot(seq(1,nrow(df.randombond),by=1),df.randombond$YtM.Out,"l")

DP

**DP (dirty price calculation of a fixed-coupon bond)**

Description

DP returns a bond’s temporal and pecuniary characteristics on the desired calendar date according to the methodology presented in Djatschenko (2018).

Usage

DP(
  CP = as.numeric(NA),
  SETT = as.Date(NA),
  Em = as.Date(NA),
  Mat = as.Date(NA),
  CpY = as.numeric(NA),
  FIPD = as.Date(NA),
  LIPD = as.Date(NA),
  FIAD = as.Date(NA),
  RV = as.numeric(NA),
  Coup = as.numeric(NA),
  DCC = as.numeric(NA),
  EOM = as.numeric(NA),
  DateOrigin = as.Date("1970-01-01"),...
InputCheck = 1,
FindEOM = FALSE,
RegCF.equal = 0,
AnnivDatesOutput = as.list(NA)
)

Arguments

CP The bond’s clean price.
SETT The settlement date. Date class object with format "%Y-%m-%d". (required)
Em The bond’s issue date. Date class object with format "%Y-%m-%d". (required)
Mat So-called "maturity date" i.e. date on which the redemption value and the final
interest are paid. Date class object with format "%Y-%m-%d". (required)
CpY Number of interest payments per year (non-negative integer; element of the set
{0,1,2,3,4,6,12}. Default: 2.
FIPD First interest payment date after Em. Date class object with format "%Y-%m-
%d". Default: NA.
LIPD Last interest payment date before Mat. Date class object with format "%Y-%m-
%d". Default: NA.
FIAD Date on which the interest accrual starts (so-called "dated date"). Date class
object with format "%Y-%m-%d". Default: NA.
RV The redemption value of the bond. Default: 100.
Coup Nominal interest rate per year in percent. Default: NA.
DCC The day count convention the bond follows. Default: NA. For a list of day count
conventions currently implemented type View(List.DCC).
EOM Boolean indicating whether the bond follows the End-of-Month rule. Default:
NA.
DateOrigin Determines the starting point for the daycount in "Date" objects. Default: "1970-
01-01".
InputCheck If 1, the input variables are checked for the correct format. Default: 1.
FindEOM If TRUE, EOM is overridden by the value inferred from the data. Default: FALSE.
RegCF.equal If 0, the amounts of regular cash flows are calculated according to the stipulated
DCC. Any other value forces all regular cash flows to be equal sized. Default: 0.
AnnivDatesOutput A list containing the output of the function AnnivDates. Default: NA.

Details

The function DP generates a list of the two data frames Dates and Cash, which contain the relevant
date-related and pecuniary characteristics that were either provided by the user or calculated by
the function. Value provides further information on the output.
Value

**Dates (data frame)**

- **Previous_CouponDate**
- **SettlementDate**
- **Next_CouponDate**

**DaysAccrued**  The number of days accrued from `Previous_CouponDate` to `Next_CouponDate`, incl. the earlier and excl. the later date.

**DaysInPeriod**  The number of interest accruing days in the coupon period from `Previous_CouponDate` to `Next_CouponDate`.

**Cash (data frame)**

- **Dirty_Price**  Sum of `Clean_Price` and `Accrued_Interest`.
- **Clean_Price**  The clean price entered.
- **Accrued_Interest**  The amount of accrued interest on `SettlementDate`.
- **CouponPayment**  The interest payment on `Next_CouponDate`.

References


Examples

```r
CP<-rep(100,16)
SETT<-rep(as.Date("2014-10-15"),16)
Em<-rep(as.Date("2013-11-30"),16)
Mat<-rep(as.Date("2021-04-21"),16)
CpY<-rep(2,16)
FIPD<-rep(as.Date("2015-02-28"),16)
LIPD<-rep(as.Date("2020-02-29"),16)
FIAD<-rep(as.Date("2013-11-30"),16)
RV<-rep(100,16)
Coup<-rep(5.25,16)
DCC<-seq(1,16,by=1)
DP.DCC_Comparison<-data.frame(CP,SETT,Em,Mat,CpY,FIPD,LIPD,FIAD,RV,Coup,DCC)

# you can pass an array to AnnivDates
List<-suppressWarnings(
  AnnivDates(unlist(DP.DCC_Comparison[1,c(3:11)],use.names=FALSE))
)

# and use its output in DP
suppressWarnings(
  DP(unlist(DP.DCC_Comparison[1,c(1:11)],use.names=FALSE),AnnivDatesOutput=List)
)

# or just apply DP to the data frame
DP.Output<-suppressWarnings(
  apply(DP.DCC_Comparison[,c("CP","SETT","Em","Mat","CpY","FIPD","LIPD","FIAD","RV","Coup","DCC")],
         1,function(y) DP(y[1],y[2],y[3],y[4],y[5],y[6],y[7],
```

```r
```
List.DCC

List of the day count conventions implemented.

Description

List of the day count conventions implemented.

Usage

data(List.DCC)

Format

A data frame with 16 rows and 3 variables:

- **DCC**  Identifier.
- **DCC.Name**  Names of the day count convention.
- **DCC.Reference**  Reference.

References

11. SWX Swiss Exchange and D. Christie, 2003, "Accrued Interest & Yield Calculations and Determination of Holiday Calendars".

NonBusDays.Brazil

Non-business days in Brazil from 1946-01-01 to 2299-12-31.

Description
This data frame contains all Saturdays and Sundays and the following Brazilian national holidays:

- New Year’s Day (always on 01. Jan)
- Shrove Monday (variable date)
- Shrove Tuesday (variable date)
- Good Friday (variable date)
- Tiradentes’ Day (always on 21. Apr)
- Labour Day (always on 01. May)
- Corpus Christi (variable date)
- Independence Day (always on 07. Sep)
- Our Lady of Aparecida (always on 12. Oct)
- All Souls’ Day (always on 02. Nov)
- Republic Day (always on 15. Nov)
- Christmas Day (always on 25. Dec)

Usage
data(NonBusDays.Brazil)

Format
A data frame with 40378 rows and 3 variables:

- Holiday.Name
- Date
- Weekday

References
PanelSomeBonds2016 A panel of 100 plain vanilla fixed coupon corporate bonds.

Description

A simulated dataset of 100 plain vanilla fixed coupon corporate bonds issued in 2016.

Usage

data(PanelSomeBonds2016)

Format

A data frame with 12718 rows and 16 variables:

ID.No  Identification number of the security.
Coup.Type  Type of the bond’s coupon.
Issue.Date  The bond’s issue date. Object of class Date with format "%Y-%m-%d".
FIAD.Input  Date on which the interest accrual starts (so-called "dated date"). Object of class Date with format "%Y-%m-%d".
FIPD.Input  First interest payment date after Issue.Date. Object of class Date with format "%Y-%m-%d".
LIPD.Input  Last interest payment date before Mat.Date. Object of class Date with format "%Y-%m-%d".
Mat.Date  So-called "maturity date" i.e. date on which the redemption value and the final interest are paid. Object of class Date with format "%Y-%m-%d".
CpY.Input  Number of interest payments per year. Object of class numeric.
Coup.Input  The nominal interest p.a. of the bond in percent. Object of class numeric.
RV.Input  The face value (= redemption value, par value) of the bond in percent.
DCC.Input  The day count convention the bond follows. Type ?AccrInt for details.
EOM.Input  Boolean indicating whether the bond follows the End-of-Month rule.
TradeDate  The calendar date on which the clean price was observed.
SETT  The settlement date that corresponds to TradeDate.
CP.Input  The clean price of the bond on TradeDate.
YtM.Input  The annualized yield to maturity of the bond on TradeDate.
Properties of 100 plain vanilla fixed coupon corporate bonds.

Description

A simulated dataset of 100 plain vanilla fixed coupon corporate bonds issued in 2016.

Usage

data(SomeBonds2016)

Format

A data frame with 100 rows and 12 variables:

- **ID.No**: Identification number of the security.
- **Coup.Type**: Type of the bond’s coupon.
- **Issue.Date**: The bond’s issue date. Object of class Date with format "%Y-%m-%d".
- **FIAD.Input**: Date on which the interest accrual starts (so-called "dated date"). Object of class Date with format "%Y-%m-%d".
- **FIPD.Input**: First interest payment date after Issue.Date. Object of class Date with format "%Y-%m-%d".
- **LIPD.Input**: Last interest payment date before Mat.Date. Object of class Date with format "%Y-%m-%d".
- **Mat.Date**: So-called "maturity date" i.e. date on which the redemption value and the final interest are paid. Object of class Date with format "%Y-%m-%d".
- **CpY.Input**: Number of interest payments per year. Object of class numeric.
- **Coup.Input**: The nominal interest p.a. of the bond in percent. Object of class numeric.
- **RV.Input**: The face value (= redemption value, par value) of the bond in percent.
- **DCC.Input**: The day count convention the bond follows. Type ?AccrInt for details.
- **EOM.Input**: Boolean indicating whether the bond follows the End-of-Month rule.
Index

* datasets
  List.DCC, 28
  NonBusDays.Brazil, 29
  PanelSomeBonds2016, 30
  SomeBonds2016, 31

  AccrInt, 2
  AnnivDates, 9

  BondVal.Price, 20
  BondVal.Yield, 22

  DP, 25

  List.DCC, 28
  NonBusDays.Brazil, 29
  PanelSomeBonds2016, 30
  SomeBonds2016, 31