

# Package ‘RcppQuantuccia’

October 30, 2021

**Type** Package

**Title** R Bindings to the Calendaring Functionality of 'QuantLib'

**Version** 0.1.0

**Date** 2021-10-30

**Author** Dirk Eddelbuettel; the authors and contributors of QuantLib

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

**Description** 'QuantLib' bindings are provided for R using 'Rcpp' via an updated variant of the header-only 'Quantuccia' project (put together initially by Peter Caspers) offering an essential subset of 'QuantLib' (and now maintained separately for the calendaring subset). See the included file 'AUTHORS' for a full list of contributors to both 'QuantLib' and 'Quantuccia'.

**URL** <https://github.com/eddelbuettel/rcppquantuccia>,  
<https://dirk.eddelbuettel.com/code/rcpp.quantuccia.html>

**BugReports** <https://github.com/eddelbuettel/rcppquantuccia/issues>

**License** GPL (>= 2)

**Imports** Rcpp

**LinkingTo** Rcpp, BH

**RoxygenNote** 6.0.1

**NeedsCompilation** yes

**Encoding** UTF-8

**Repository** CRAN

**Date/Publication** 2021-10-30 14:30:05 UTC

## R topics documented:

RcppQuantuccia-package . . . . .	2
adjust_cpp . . . . .	3
advanceDate . . . . .	4
advanceUnits_cpp . . . . .	5

businessDaysBetween . . . . .	6
calendars . . . . .	6
getEndOfMonth . . . . .	7
getHolidays . . . . .	7
getName . . . . .	8
isBusinessDay . . . . .	9
isEndOfMonth . . . . .	9
isHoliday . . . . .	10
isWeekend . . . . .	11
setCalendar . . . . .	11

<b>Index</b>	<b>13</b>
--------------	-----------

---

RcppQuantuccia-package

*R Bindings to the Calendaring Functionality of 'QuantLib'*

---

## Description

'QuantLib' bindings are provided for R using 'Rcpp' via an updated variant of the header-only 'Quantuccia' project (put together initially by Peter Caspers) offering an essential subset of 'QuantLib' (and now maintained separately for the calendaring subset). See the included file 'AUTHORS' for a full list of contributors to both 'QuantLib' and 'Quantuccia'.

## Details

The DESCRIPTION file:

```

Package:      RcppQuantuccia
Type:         Package
Title:        R Bindings to the Calendaring Functionality of 'QuantLib'
Version:      0.1.0
Date:         2021-10-30
Author:       Dirk Eddelbuettel; the authors and contributors of QuantLib
Maintainer:   Dirk Eddelbuettel <edd@debian.org>
Description:  'QuantLib' bindings are provided for R using 'Rcpp' via an updated variant of the header-only 'Quantuccia' project (put together initially by Peter Caspers) offering an essential subset of 'QuantLib' (and now maintained separately for the calendaring subset). See the included file 'AUTHORS' for a full list of contributors to both 'QuantLib' and 'Quantuccia'.
URL:          https://github.com/eddelbuettel/rcppquantuccia, https://dirk.eddelbuettel.com/code/rcpp.quantuccia.html
BugReports:   https://github.com/eddelbuettel/rcppquantuccia/issues
License:      GPL (>= 2)
Imports:      Rcpp
LinkingTo:    Rcpp, BH
RoxygenNote: 6.0.1
NeedsCompilation: yes
Encoding:     UTF-8

```

**Package Content**

Index of help topics:

RcppQuantuccia-package

R Bindings to the Calendaring Functionality of 'QuantLib'

adjust\_cpp            Compute adjusted dates

advanceDate         Advance a date

advanceUnits\_cpp    Compute adjusted dates

businessDaysBetween    Compute number of business dates between calendar dates

calendars            The 'calendars' vector contains all calendar identifiers.

getEndOfMonth        Compute end-of-month

getHolidays         Compute holidays or business days

getName              Get calendar name, or id

isBusinessDay        Test for business days

isEndOfMonth        Test for end-of-month

isHoliday            Test for holidays

isWeekend            Test for weekends

setCalendar          Set a calendar

**Maintainer**

Dirk Eddelbuettel <edd@debian.org>

**Author(s)**

Dirk Eddelbuettel; the authors and contributors of QuantLib

**References**

<https://www.quantlib.org/>

---

adjust_cpp	<i>Compute adjusted dates</i>
------------	-------------------------------

---

**Description**

Adjust a vector of dates following a business-day convention

**Usage**

```
adjust_cpp(dates, bdc = 0L)
```

```
adjust(dates, bdc = c("Following", "ModifiedFollowing", "Preceding",
  "ModifiedPreceding", "Unadjusted", "HalfMonthModifiedFollowing", "Nearest"))
```

**Arguments**

dates	A Date vector with dates
bdc	A character variable describing one of several supported values, the C++ version implements expects a corresponding integer value

**Details**

This function takes a vector of dates and returns another vector of dates of the same length returning at each position the adjusted date according to the selected business-day convention. Currently supported values for the business day convention are (starting from zero): 'Following', 'ModifiedFollowing', 'Preceding', 'ModifiedPreceding', 'Unadjusted', 'HalfModifiedFollowing' and 'Nearest'.

**Value**

A Date vector with dates adjust according to business-day convention

**Examples**

```
adjust(Sys.Date()+0:6)
```

---

advanceDate	<i>Advance a date</i>
-------------	-----------------------

---

**Description**

Advance a date to the next business day plus an optional shift

**Usage**

```
advanceDate(rd, days = 0L)
```

**Arguments**

rd	A Date object describing the date to be advanced to the next business day.
days	An optional integer offset applied to the date

**Details**

This function takes a given date and advances it to the next business day under the current (global) calendar setting. If an optional offset value is given it is applied as well.

**Value**

The advanced date is returned

**Examples**

```
advanceDate(Sys.Date(), 2) # today to the next biz day, plus 2 days
```

---

advanceUnits\_cpp      *Compute adjusted dates*

---

### Description

Advance a vector of dates by a given number of time units

### Usage

```
advanceUnits_cpp(dates, n, unit, bdc, emr)
```

```
advanceUnits(dates, n, unit = c("Days", "Weeks", "Months", "Years", "Hours",
  "Minutes", "Seconds", "Milliseconds", "Microseconds"), bdc = c("Following",
  "ModifiedFollowing", "Preceding", "ModifiedPreceding", "Unadjusted",
  "HalfMonthModifiedFollowing", "Nearest"), emr = FALSE)
```

### Arguments

dates	A Date vector with dates
n	An integer variable with the number of units to advance
unit	A character variable describing one of several supported values; the C++ version implements expects a corresponding integer value
bdc	A character variable describing one of several supported values, the C++ version implements expects a corresponding integer value
emr	A boolean variable select end-of-month, default is 'FALSE'

### Details

This function takes a vector of dates and returns another vector of dates of the same length returning at each position the date advanced by the given number of steps in the selected time unit, also respecting a business day convention and and of month boolean switch. Currently supported values for the time unit are 'Days', 'Weeks', 'Months' 'Years', 'Hours', 'Seconds', 'Milliseconds' and 'Microseconds'; all are specified as integers. Note that intra-daily units are not currently supported for advancing 'Date' objects. Currently supported values for the business day convention are (starting from zero): 'Following', 'ModifiedFollowing', 'Preceding', 'ModifiedPreceding', 'Unadjusted', 'HalfModifiedFollowing' and 'Nearest'.

### Value

A Date vector with dates advanced according to the selected inputs

### Examples

```
advanceUnits(Sys.Date()+0:6, 5, "Days", "Following")
```

---

businessDaysBetween    *Compute number of business dates between calendar dates*

---

**Description**

Compute the number of business days between dates

**Usage**

```
businessDaysBetween(from, to, includeFirst = TRUE, includeLast = FALSE)
```

**Arguments**

from	A Date vector with interval start dates
to	A Date vector with interval end dates
includeFirst	A boolean indicating if the start date is included, default is 'TRUE'
includeLast	A boolean indicating if the end date is included, default is 'FALSE'

**Details**

This function takes two vectors of start and end dates and returns another vector of the number of business days between each corresponding date pair according to the active calendar.

**Value**

A numeric vector with the number of business dates between the corresponding date pair

**Examples**

```
businessDaysBetween(Sys.Date() + 0:6, Sys.Date() + 3 + 0:6)
```

---

calendars    *The calendars vector contains all calendar identifiers.*

---

**Description**

The calendars vector contains all calendar identifiers.

**Examples**

```
head(calendars, 10)
```

---

`getEndOfMonth`                      *Compute end-of-month*

---

**Description**

Compute a vector of dates with end-of-month

**Usage**

`getEndOfMonth(dates)`

**Arguments**

`dates`                      A Date vector with dates

**Details**

This function takes a vector of dates and returns another vector of dates of the same length returning at each position whether the corresponding end-of-month date in the currently active (global) calendar.

**Value**

A Date vector with dates which are end-of-month

**Examples**

`getEndOfMonth(Sys.Date()+0:6)`

---

`getHolidays`                      *Compute holidays or business days*

---

**Description**

Compute the number of holidays (or business days) between two dates

**Usage**

`getHolidays(from, to, includeWeekends = FALSE)`

`getBusinessDays(from, to)`

**Arguments**

from	A Date object with the start date
to	A Date object with the end date
includeWeekends	A boolean indicating if weekends should be included, default is 'FALSE'

**Details**

This function takes a start and end date and returns a vector of holidays (or business days) between them according to the active calendar.

**Value**

A Date vector with holidays or business days between the given dates

**Examples**

```
getHolidays(Sys.Date(), Sys.Date() + 30)
```

---

getName	<i>Get calendar name, or id</i>
---------	---------------------------------

---

**Description**

Get calendar name or id

**Usage**

```
getName()
```

```
getId()
```

**Details**

This function returns the corresponding (full) name (as in the underlying implementation class) or identification string (used to select it) of the current calendar.

**Value**

A string with the calendar name

**Examples**

```
getName()
```



---

isBusinessDay	<i>Test for business days</i>
---------------	-------------------------------

---

**Description**

Test a vector of dates for business day

**Usage**

```
isBusinessDay(dates)
```

**Arguments**

dates	A Date vector with dates to be examined
-------	---

**Details**

This function takes a vector of dates and returns a logical vector of the same length indicating at each position whether the corresponding date is a business day in the currently active (global) calendar.

**Value**

A logical vector indicating which dates are business days

**Examples**

```
isBusinessDay(Sys.Date()+0:6)
```

---

isEndOfMonth	<i>Test for end-of-month</i>
--------------	------------------------------

---

**Description**

Test a vector of dates for end-of-month

**Usage**

```
isEndOfMonth(dates)
```

**Arguments**

dates	A Date vector with dates to be examined
-------	---

**Details**

This function takes a vector of dates and returns a logical vector of the same length indicating at each position whether the corresponding date is at the end of a month in the currently active (global) calendar.

**Value**

A logical vector indicating which dates are end-of-month

**Examples**

```
isEndOfMonth(Sys.Date()+0:6)
```

---

isHoliday

*Test for holidays*

---

**Description**

Test a vector of dates for holiday

**Usage**

```
isHoliday(dates)
```

**Arguments**

dates            A Date vector with dates to be examined

**Details**

This function takes a vector of dates and returns a logical vector of the same length indicating at each position whether the corresponding date is a holiday in the currently active (global) calendar.

**Value**

A logical vector indicating which dates are holidays

**Examples**

```
isHoliday(Sys.Date()+0:6)
```

---

isWeekend	<i>Test for weekends</i>
-----------	--------------------------

---

**Description**

Test a vector of dates for weekends

**Usage**

```
isWeekend(dates)
```

**Arguments**

dates            A Date vector with dates to be examined

**Details**

This function takes a vector of dates and returns a logical vector of the same length indicating at each position whether the corresponding date is a weekend in the currently active (global) calendar.

**Value**

A logical vector indicating which dates are weekends

**Examples**

```
isWeekend(Sys.Date()+0:6)
```

---

setCalendar	<i>Set a calendar</i>
-------------	-----------------------

---

**Description**

Set a calendar

**Usage**

```
setCalendar(calstr)
```

**Arguments**

calstr            A character variable containing the market for which a calendar is to be set

**Details**

This function sets a calendar to the given market or country convention. Note that at present only the default 'TARGET' and 'UnitedStates' are supported.

**Value**

Nothing is returned but the global state is changed

**Examples**

```
setCalendar("UnitedStates")
```

# Index

- \* **data**

- calendars, [6](#)

- \* **package**

- RcppQuantuccia-package, [2](#)

adjust (adjust\_cpp), [3](#)

adjust\_cpp, [3](#)

advanceDate, [4](#)

advanceUnits (advanceUnits\_cpp), [5](#)

advanceUnits\_cpp, [5](#)

businessDaysBetween, [6](#)

calendars, [6](#)

getBusinessDays (getHolidays), [7](#)

getEndOfMonth, [7](#)

getHolidays, [7](#)

getId (getName), [8](#)

getName, [8](#)

isBusinessDay, [9](#)

isEndOfMonth, [9](#)

isHoliday, [10](#)

isWeekend, [11](#)

RcppQuantuccia

(RcppQuantuccia-package), [2](#)

RcppQuantuccia-package, [2](#)

setCalendar, [11](#)