Package 'datetimeutils'

June 30, 2022

Type Package

Title Utilities for Dates and Times

Version 0.6-0

Date 2022-05-27

Maintainer Enrico Schumann <es@enricoschumann.net>

Description Utilities for handling dates and times, such as selecting particular days of the week or month, formatting timestamps as required by RSS feeds, or converting timestamp representations of other software (such as 'MATLAB' and 'Excel') to R. The package is lightweight (no dependencies, pure R implementations) and relies only on R's standard classes to represent dates and times ('Date' and 'POSIXt'); it aims to provide efficient implementations, through vectorisation and the use of R's native numeric representations of timestamps where possible.

Suggests tinytest

License GPL-3

URL http://enricoschumann.net/R/packages/datetimeutils/, https://github.com/enricoschumann/datetimeutils

LazyData yes

NeedsCompilation no

Author Enrico Schumann [aut, cre] (<https://orcid.org/0000-0001-7601-6576>), Unicode, Inc. [dtc, cph] (source of timezone names in 'tznames')

R topics documented:

`datetimeutils-package` *Utilities for Dates and Times*

Description

Utilities for handling dates and times, such as selecting particular days of the week or month, formatting timestamps as required by RSS feeds, or converting timestamp representations of other software (such as 'MATLAB' and 'Excel') to R. The package is lightweight (no dependencies, pure R implementations) and relies only on R's standard classes to represent dates and times ('Date' and 'POSIXt'); it aims to provide efficient implementations, through vectorisation and the use of R's native numeric representations of timestamps where possible.

Details

Helper functions for dealing with times and dates.

Author(s)

Enrico Schumann

Maintainer: Enrico Schumann <es@enricoschumann.net>

References

B.D. Ripley and K. Hornik. *Date-Time Classes*. R-News, **1**(2):8–12, 2001.

See Also

`DateTimeClasses`, `Dates`

business_days *Business Days*

Description

Check whether a timestamp of class Date or POSIXt is a business day; compute past or future business days.

Usage

```
is_businessday(x, holidays = NULL)
is_weekend(x)
previous_businessday(x, holidays = NULL, shift = -1)
prev_bday(x, holidays = NULL, shift = -1)
next_businessday(x, holidays = NULL, shift = 1)
next_bday(x, holidays = NULL, shift = 1)
```

Arguments

x	a vector of class Date or POSIXct
holidays	A vector of class Date, or a character vector in a format that is understood by as.Date, or anything that can be coerced to class Date by as.Date (e.g. POSIXt).
shift	integer

Details

is_weekend checks whether a given date is a Saturday or Sunday.

previous_businessday takes a Date x and returns the last non-weekend day before. When shift is less than -1, the function evaluates to the shift-th previous day. When shift is 0, the function will return x if it is a business day, else the previous business day. next_businessday works analogously. There are shorter-named versions next_bday and prev_bday.

Value

Logical.

Author(s)

Enrico Schumann

References

B.D. Ripley and K. Hornik. *Date-Time Classes*. R-News, **1**(2):8–12, 2001.

See Also

DateTimeClasses

Examples

```
is_weekend(Sys.Date())
previous_businessday(Sys.Date())
next_businessday(Sys.Date())
```

---

convert_date *Convert Various Formats to Date*

Description

Convert dates in external formats (e.g. from MATLAB) to Date or POSIXct.

Usage

```
convert_date(x, type, fraction = FALSE, tz = "")
```

Arguments

x	numeric
type	character: "excel", "excel1904", "matlab" and "spss"/"pspp" are supported.
fraction	logical: should fractional dates (i.e. times) be used? Default is FALSE.
tz	character: if fraction is TRUE, then what time zone is to be assumed? Default is "", i.e. the local time zone.

Details

Convert the numeric representation of a date to class Date. Note that different versions of Excel use different origins: 1900-01-01 or 1904-01-01. For the latter, set type to "excel1904". For the former, convert_date uses 1899-12-31 because Excel considers 1900 a leap year (which it is not). So dates before 1 March 1900 are probably wrong (off by one day).

Value

A vector of class Date, or POSIXct if fraction is TRUE.

Author(s)

Enrico Schumann; type spss/pspp suggested and based on a patch by J\"org Beyer

See Also

as.Date, as.POSIXlt

Examples

```
convert_date(40000, "excel")
```

`convert_tz` *Convert a Timestamp from one Timezone to Another*

Description

Convert a timestamp from one timezone to another.

Usage

```
convert_tz(datetime, from = "", to)
```

Arguments

`datetime`	character: `YYYY-MM-DD HH:MM:SS`
`from`	the timezone of datetime. If `""`, the local timezone is used.
`to`	to timezone to which datetime should be converted

Details

See `timezones`.

Be careful: if the specified timezone does not exist on your system, the function will **not** return an error.

Value

`POSIXct`

Author(s)

Enrico Schumann

References

B.D. Ripley and K. Hornik. *Date-Time Classes*. R-News, **1**(2):8–12, 2001.

See Also

`POSIXct`

Examples

```
convert_tz("2016-05-10 12:00:00",
          "America/Chicago", "America/Chicago")

convert_tz("2016-05-10 12:00:00",
          "Europe/Berlin", "America/Chicago")

convert_tz(Sys.time(), to = "Europe/London")
convert_tz(Sys.time(), to = "America/Chicago")
```

---

date1904 *Is File Origin 1904?*

Description

Checks whether an xlsx file uses 1 Jan 1904 as its origin.

Usage

```
date1904(filename)
```

Arguments

filename	character: one or more filenames

Details

Requires `utils::unzip`.

Value

A logical vector: `TRUE` if origin is 1904; `FALSE` if origin is 1900; `NA` if file could not be processed.

Author(s)

Enrico Schumann; type spss/pspp suggested and based on a patch by J\"org Beyer

References

ECMA-376-1:2016 *Office Open XML File Formats*.

See Also

convert_date

Examples

```
date1904("~/Desktop/02_company_statistics.pdf")
```

---

end_of_period *Handling and Manipulating Dates and Times*

Description

Functions for manipulating vectors that inherit from class `POSIXt` or `Date`.

Usage

```
is_leapyear(x)

first_of_month(x)
end_of_month(x, shift = 0L)
end_of_previous_month(x)

first_of_year(x)
end_of_year(x, shift = 0L)
end_of_previous_year(x)

end_of_quarter(x, shift = 0L)

day_of_month(x)
day_of_month(x) <- value
mday(x)
mday(x) <- value

second(x, as.character = FALSE)
minute(x, as.character = FALSE)
hour(x, as.character = FALSE)
month(x, as.character = FALSE)
year(x, as.character = FALSE)
```

Arguments

`x`	a vector of class `Date`
`value`	a vector of integers
`shift`	integer
`as.character`	logical

Details

`end_of_month` returns the last calendar day of a given month. If `shift` is positive, then shift `months` into the future; if negative, the end of previous months. `end_of_month(x,-1)` is equivalent to `end_of_previous_month(x)`. `end_of_year` works in the same way, but for calendar years.

`mday` is a wrapper for `day_of_month`.

Value

Vectors of class `Date` or `POSIXct`; or `logical`

Author(s)

Enrico Schumann

References

B.D. Ripley and K. Hornik. *Date-Time Classes*. R-News, **1**(2):8–12, 2001.

See Also

`DateTimeClasses`

Many useful functions are also in package **chron**.

Examples

```
## vectorisation: x *or* shift (but not both!)
end_of_month(as.Date("2013-01-15"), shift = c(-1, 0, 1))
end_of_month(as.Date("2013-01-15") + 0:100)


day_of_month(d <- as.Date("2013-01-22"))
day_of_month(d) <- 5
d
```

`guess_datetime` *Guess Timestamp Format*

Description

Tries to convert a character vector to POSIXct.

Usage

```
guess_datetime(s, date.only = FALSE, within = FALSE, tz = "",
               try.patterns = NULL)
```

Arguments

s	character
date.only	logical: try to guess dates only (if TRUE) or times as well (if FALSE)
within	logical: ignore surrounding text? Note that trailing text is always ignored, see as.Date.
tz	character: timezone to assume for times. Default is the current timezone. See argument tz in as.POSIXct
try.patterns	either NULL or a character vector. See Details and Examples.

Details

The function first coerces its argument to character. It then applies a list of patterns to each element of s. Let d be a numeric digit; then the rules are roughly those in the table below. (For the precise rules, see Examples below.)

original pattern	*assumed format*
dddd-dd-dd dd:dd:dd	%Y-%m-%d %H:%M:%S
dd/dd/dddd dd:dd:dd	%m/%d/%Y %H:%M:%S
dd.dd.dddd dd:dd:dd	%d.%m.%Y %H:%M:%S

The rules are followed in the given order; an element will be matched only once. If there is a match, strptime will be tried with the *assumed format* (when date.only is TRUE, as.Date will be tried). For elements that do not match any pattern or for which strptime fails, NA is returned.

Additional patterns can be specified as try.patterns. This must be a character vector with an even number of elements: the first of each pair of elements is used as the pattern in a regular expression; the second as the format string passed to strptime. See Examples.

Value

POSIXct

Warning

If you know the format of a timestamp, then **do not use this function** (use strptime instead). If you have no idea at all about the format of a timestamp, then **do not use this function**.

Author(s)

Enrico Schumann

See Also

strptime

Examples

```
s <- c("  1999-08-19     10:00:31   ",
       "   1999-08-19 10:00",
       "19.8.1999 10:00",
       "8/19/99      10:00:31",
       "8/19/1999 10:00:31",
       "19.8.1999 10:00:31")

guess_datetime(s)

## the actual rules
rules <- as.data.frame(matrix(datetimeutils:::.dt_patterns,
                              byrow = TRUE, ncol = 2),
                       stringsAsFactors = FALSE)
names(rules) <- c("pattern", "assumed_format")
rules



## ---------------------------------

## a function for finding old files by looking at the
## dates in filenames (e.g. in a backup directory)
old_files <- function(min.age = 365,   ## in days
                      path = ".",
                      recursive = FALSE,
                      full.names = FALSE) {

    files <- dir(path, recursive = recursive, full.names = full.names)
    dates <- guess_datetime(files, date.only = TRUE, within = TRUE)

    age <- as.numeric(Sys.Date() - dates)
    old <- age >= min.age

    files[ !is.na(old) & old ]
}


## ---------------------------------

## specifying additional formats

s <- c("19-08-99",
       "29-2-00")
guess_datetime(s, date.only = TRUE)
## NA NA
guess_datetime(s, date.only = TRUE,
               try.patterns = c("[0-9]+-[0-9]+-[0-9]+", "%d-%m-%y"))
## "1999-08-19" "2000-02-29"
```

last_weekday *Functions for Computing Days of the Week*

Description

Functions for computing a specifed day-of-week, such as 'the last Friday of October 2015'.

Usage

```
last_weekday(weekday, x, shift = 0L,
             period = "month", before, inclusive = TRUE)
nth_weekday(weekday, x, n = 1L)
```

Arguments

x	a vector of class Date (but only the YYYY-MM part is relevant)
shift	a vector of integers
weekday	an integer (1 is Monday, 2 is Tuesday, and so on)
period	character. Currently ignored.
before	a Date. See also inclusive.
inclusive	logical. Is before meant is 'before but including'?
n	an integer

Details

last_weekday computes the last day-of-the-week (specified as an integer 0 to 6, with Sunday being 0) in a given month, e.g. 'the last Friday'. shift moves forward (when positive) or backward (when negative) by one week; see Examples.

nth_weekday gives the *n*-th day-of-the-week (specified as an integer 0 to 6, with Sunday being 0) of a given month, e.g. 'the second Monday'.

Value

Date

Author(s)

Enrico Schumann

References

B.D. Ripley and K. Hornik. *Date-Time Classes*. R-News, **1**(2):8–12, 2001.

See Also

DateTimeClasses

Many useful functions are also in package **chron**.

Examples

```
## GOAL:     find the third Friday in March 2013
## SOLUTION: find the last Friday in February 2013 and
##           shift forward by 3 weeks
last_weekday(5, as.Date("2013-02-01"), shift = 3)

## ... or much simpler
nth_weekday(5, as.Date("2013-03-01"), 3)
```

`month.name.de` *Non-English Month Names and Abbreviations*

Description

Month names and abbreviations in languages other than English: Currently only German is supported.

Usage

```
month.name.de
month.abb.de.din1355.1
```

Format

Character vectors; encoded as UTF-8 if necessary.

Details

Character vectors, encoded as UTF-8.

`month.abb.de.din1355.1` contains the abbreviations of the withdrawn DIN 1355-1, which uses "Mrz" for March.

Source

https://de.wikipedia.org/wiki/DIN_1355-1

References

https://de.wikipedia.org/wiki/DIN_1355-1

Examples

```
month.name.de
month.name.de[month(Sys.Date())]
```

nth_day *Compute Reference Dates*

Description

Compute sequences of reference dates, such as last day of month or first day of quarter.

Usage

```
nth_day(timestamps, period = "month", n,
        start, end, business.days = FALSE,
        missing = "previous", index = FALSE)
```

Arguments

timestamps	timestamps: a sorted vector of Dates
period	numeric or character: supported are "week", "month", "quarter", "halfyear", "year". If numeric, period is interpreted as a month number, with January being 1. Also possible are month names, either English as in month.name or month.abb, or as defined in the current locale (see strftime format specification "%b" and "%B").
n	numeric or character: currently supported are "first" and "last". If numeric, it will be interpreted as the *n*-th day of the period.
start	Date
end	Date
business.days	logical
missing	character. Not supported yet.
index	logical. If TRUE, the indices (instead of actual timestamps) are returned.

Details

The function computes sequences of dates that are often used as reference dates, for instance in financial reporting: last day of the month or of the year, or a particular day of the month.

The function takes a vector of timestamps and returns a subset of these timestamps. Alternatively, a sequence of calendar days may be constructed by specifying start and end.

Value

A vector of timestamps or, if index is TRUE, a vector of integers.

Author(s)

Enrico Schumann

See Also

nth_weekday

Examples

```
timestamps <- seq(from = as.Date("2001-01-01"),
                  to   = as.Date("2001-04-15"),
                  by   = "1 day")

nth_day(timestamps, period = "quarter", n = "last")
## [1] "2001-03-31" "2001-04-15"

nth_day(timestamps, period = "quarter", n = 10)
## [1] "2001-01-10" "2001-04-10"

nth_day(timestamps, period = "quarter", n = 1:2)
## [1] "2001-01-01" "2001-01-02" "2001-04-01" "2001-04-02"

nth_day(timestamps, period = "month", n = "last")
## [1] "2001-01-31" "2001-02-28" "2001-03-31" "2001-04-15"


nth_day(start = as.Date("2016-06-03"),
        end   = as.Date("2017-08-01"),
        period = c(6, 12), n = 3)
## [1] "2016-06-05" "2016-12-03" "2017-06-03"

nth_day(start = as.Date("2016-06-03"),
        end   = as.Date("2017-08-01"),
        period = c("Jun", "Dec"), n = c(3, 5))
## [1] "2016-06-05" "2016-06-07" "2016-12-03" "2016-12-05"
## [5] "2017-06-03" "2017-06-05"
```

---

rfc822t *Format Date and Time as Described in RFC 822*

Description

Format a timestamp as described in RFC 822.

Usage

```
rfc822t(x, include.dow = TRUE)
```

Arguments

x	a vector that can be coerced to POSIXlt
include.dow	logical; include the day of the week?

Details

Formats a timestamp as '%Y %H:%M:%S %z', possibly prepending an abbreviated day-of-week. The function ignores the current locale: day-of-week and month names are in English. The format is required for timestamps in RSS feeds.

Value

a character vector

Author(s)

Enrico Schumann

References

https://www.ietf.org/rfc/rfc0822.txt

https://www.rssboard.org/rss-specification

See Also

strftime, date

Examples

```
rfc822t(Sys.time())
```

---

roundPOSIXt *Round POSIXt Objects to Specified Interval*

Description

Round POSIXt objects to specified intervals such as '5 minutes'.

Usage

```
roundPOSIXt(t, interval, up = FALSE)
```

Arguments

t	a vector that inherits from class POSIXt
interval	A character string of the form "num units", in which num is a number, and units is sec, min, hour or day. num and units must be separated by white space.
up	logical: round down (the default) or up?

Details

roundPOSIXt rounds an input of class POSIXt; it returns a vector of class POSIXct.

Value

POSIXct

Author(s)

Enrico Schumann

References

B.D. Ripley and K. Hornik. *Date-Time Classes*. R-News, **1**(2):8–12, 2001.

See Also

DateTimeClasses

Examples

```
times <- as.POSIXct("2012-03-24 22:17:27") + 1:3
roundPOSIXt(times, "10 min")
roundPOSIXt(times, "10 min", TRUE)
```

---

timegrid *POSIXct Time Grid*

Description

Build an equally-spaced sequence of POSIXct timestamps.

Usage

```
timegrid(from, to, interval,
         exclude.weekends = TRUE, holidays = NULL,
         fromHHMMSS = "080000", toHHMMSS = "220000")
```

Arguments

from	a vector of length one that inherits from class POSIXt. If there from has a time-zone attribute, it will be used for the grid.
to	a vector of length one that inherits from class POSIXt
interval	A character string like "num units", in which num is a number, and units is sec, min, hour or day. num and units must be separated by white space.
exclude.weekends	logical; default is TRUE
fromHHMMSS	A character vector of length one like "HHMMSS". Times-of-day earlier than HHMMSS are excluded from the grid. The applicable timezone will be taken from the from argument.

toHHMMSS	A character vector of length one like "HHMMSS". Times-of-day later than HHMMSS are excluded from the grid. The applicable timezone will be taken from the from argument.
holidays	A vector of class Date, or a character vector in a format that is understood by as.Date, or anything that can be coerced to class Date by as.Date (eg, POSIXt).

Details

timegrid creates an equally-spaced grid of class POSIXct.

Value

a vector of class POSIXct (or a character vector of length zero, in case no valid points remain)

Author(s)

Enrico Schumann

References

B.D. Ripley and K. Hornik. *Date-Time Classes*. R-News, **1**(2):8–12, 2001.

See Also

strftime, date

Examples

```
from <- as.POSIXct("2012-04-30 08:00:00")
to   <- as.POSIXct("2012-05-04 22:00:00")
timegrid(from, to, interval = "1 hour",
         holidays = as.Date("2012-05-01"))


timegrid(as.POSIXct("2017-06-23 21:00:00"), ## system timezone
         as.POSIXct("2017-06-26 10:00:00"),
         interval = "15 min")
timegrid(as.POSIXlt("2017-06-23 21:00:00", tz = "UTC"),
         as.POSIXlt("2017-06-26 10:00:00", tz = "UTC"),
         interval = "15 min")
```

tznames *Timezone Names*

Description

A mapping between tz database (a.k.a. Olson database) and Windows timezone names.

Usage

```
data("tznames")
```

Format

A data frame of the following 2 variables:

`Windows` a character vector: the timezone names used under Windows and its applications (e.g. in Outlook calendars)

`Olson` a character vector of the names returned by `OlsonNames`

Details

The data are auto-generated from file `windowsZones.xml` in the Unicode Common Locale Data Repository (http://cldr.unicode.org/). See https://www.unicode.org/copyright.html and https://www.unicode.org/license.html for the terms of use.

There is no 1-to-1 mapping between names: several Olson names typically map to a single Windows name.

Source

Unicode Common Locale Data Repository (CLDR) http://cldr.unicode.org/

References

See https://www.iana.org/time-zones and http://web.cs.ucla.edu/~eggert/tz/tz-link.htm for more information about the tz database.

See also `OlsonNames`.

A plain-text table is at https://github.com/enricoschumann/datetimeutils/blob/master/data/tznames.txt

Examples

```
str(tznames)
```

Index