

Package ‘fastymd’

July 22, 2025

Title Fast Utilities for Year Month Day Objects

Version 0.1.3

Description A collection of utility functions for working with Year Month Day objects. Includes functions for fast parsing of numeric and character input based on algorithms described in Hinnant, H. (2021) <https://howardhinnant.github.io/date_algorithms.html> as well as a branchless calculation of leap years by Jerichaux (2025) <<https://stackoverflow.com/a/79564914>>.

License GPL-3

URL <https://timtaylor.codeberg.page/fastymd/>

BugReports <https://codeberg.org/TimTaylor/fastymd/issues>

Encoding UTF-8

Suggests fasttime, lubridate, microbenchmark, tinytest, ymd, litedown

Depends R (>= 4.2.0)

VignetteBuilder litedown

RoxygenNote 7.3.2

Config/testthat/load-all list(export_all = FALSE, helpers = FALSE)

NeedsCompilation yes

Author Tim Taylor [aut, cre] (ORCID: <<https://orcid.org/0000-0002-8587-7113>>),
Howard Hinnant [aut] (Author of underlying algorithms for calculating
Epoch days to Calendar days and vice versa),
jerichaux [aut] (Author of included branchless leap year calculation)

Maintainer Tim Taylor <tim.taylor@hidden elephants.co.uk>

Repository CRAN

Date/Publication 2025-05-12 19:50:02 UTC

Contents

accessors	2
fymd	3
is_leap_year	4

Index	6
--------------	----------

accessors

Generics for accessing the year, month and month-day of an object

Description

Fast methods are provided for Date objects. The underlying algorithm follows the approach described in Hinnant (2021) for converting days since the **UNIX Epoch** to **Gregorian Calendar** dates.

Usage

```
get_ymd(x, ...)
```

```
get_year(x, ...)
```

```
get_month(x, ...)
```

```
get_mday(x, ...)
```

Arguments

`x` An R object.

`...` Further arguments passed to or from other methods.

Value

For `get_ymd()` a data frame with integer columns `year`, `month` and `mday`. For `get_year()`, `get_month()` and `get_mday()`, integer vectors of the requested components.

References

Hinnant, J. (2021) *chrono-Compatible Low-Level Date Algorithms*. Available at: https://howardhinnant.github.io/date_algorithms.html#civil_from_days (Accessed 17 April 2025).

Examples

```
date <- as.Date("2025-04-17")
get_ymd(date)
get_year(date)
get_month(date)
get_mday(date)
```

fynd

*Construct dates from character and numeric input***Description**

`fynd()` is a generic for validated conversion of R objects to (integer) Date. Efficient methods are provided for numeric and character inputs.

Usage

```
fynd(...)

## Default S3 method:
fynd(...)

## S3 method for class 'numeric'
fynd(y, m = 1, d = 1, ...)

## S3 method for class 'character'
fynd(x, strict = FALSE, ...)
```

Arguments

<code>...</code>	Arguments to be passed to or from other methods.
<code>y, m, d</code>	integerish. Numeric vector corresponding to the desired years, months and days. Double vectors are coerced to integer. Length 1 vectors will be recycled to the common size across <code>y</code> , <code>m</code> and <code>d</code> .
<code>x</code>	character. Vector of year-month-date strings in a numeric format (e.g. "2020-02-01"). Parses digits separated by non-digits. Leading and trailing whitespace will be ignored.
<code>strict</code>	bool. Should non-whitespace output after a valid date be allowed? FALSE (default) will ignore output after a valid date whereas TRUE will reject said strings, returning NA.

Details

The underlying algorithm for both the numeric and character methods follow the approach described in Hinnant (2021) for calculating days from the [UNIX Epoch](#) from [Gregorian Calendar](#) dates.

The character version parses inputs in a fixed, year, month and day order. These values must be digits but can be separated by any non-digit character. It is similar in spirit to that of Simon Urbanek's `fastDate()` implementation in that we use pure text parsing and no system calls. `fynd()` differs from `fastDate()` in that it validates all dates for correctness and supports a much larger

range of dates (i.e. the **Proleptic Gregorian calendar**. This additional capability does come with a small performance cost but, IMO, remains competitive.

For both numeric and character versions years must be in the range [-9999, 9999].

Value

A Date object

References

Hinnant, H. (2021) *chrono-Compatible Low-Level Date Algorithms*. Available at: https://howardhinnant.github.io/date_algorithms.html#days_from_civil (Accessed 17 April 2025).

Urbanek S (2022). *fasttime: Fast Utility Function for Time Parsing and Conversion*. R package version 1.1-0, doi:10.32614/CRAN.package.fasttime.

Examples

```
cdate      <- "2025-04-16"
timestamp <- "2025-04-16T09:45:53+0000"

# Ignoring the time element
fynd(timestamp)

# This will return NA with a warning
fynd(timestamp, strict = TRUE)

# Checking
as.Date(cdate) == fynd(timestamp)

# Leap year
fynd(2020, 2, 29)

# Not a leap year
fynd(2021, 2, 29)
```

is_leap_year	<i>Is value a leap year?</i>
--------------	------------------------------

Description

Determine whether an input value is a leap year using the branchless approach of jerichaux (2025). Method provided for both Dates and numeric values. Numeric values first floored before the calculation is made.

Usage

```
is_leap_year(x)

is_leap(x)
```

Arguments

x An R object.

Value

logical result.

References

jerichaux. (2025) *How to find leap year programmatically in C*. Available at: <https://stackoverflow.com/a/79564914> (Accessed 16 April 2025).

Index

accessors, [2](#)

fastDate(), [3](#)

fynd, [3](#)

get_mday (accessors), [2](#)

get_month (accessors), [2](#)

get_year (accessors), [2](#)

get_ynd (accessors), [2](#)

is_leap (is_leap_year), [4](#)

is_leap_year, [4](#)