Package ‘dint’

February 6, 2020

Type Package

Title A Toolkit for Year-Quarter, Year-Month and Year-Isoweek Dates

Version 2.1.3

Maintainer Stefan Fleck <stefan.b.fleck@gmail.com>

Description S3 classes and methods to create and work with year-quarter, year-month and year-isoweek vectors. Basic arithmetic operations (such as adding and subtracting) are supported, as well as formatting and converting to and from standard R date types.

License MIT + file LICENSE


Suggests covr, ggplot2, knitr, lubridate, rmarkdown, scales, testthat, zoo

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 7.0.2.9000

Collate 'accessors.R' 'arithmetic.R' 'date_xx.R' 'date_y.R'
   'date_ym.R' 'date_yq.R' 'date_yw.R' 'dint-package.R'
   'extract.r' 'first_of.R' 'format.R' 'increment.R' 'parser.R'
   'predicates.R' 'utils-sfmisc.R' 'zoo-compat.R'
   'scale_date_xx.R' 'seq.R' 'utils.R'

NeedsCompilation no

Author Stefan Fleck [aut, cre] (<https://orcid.org/0000-0003-3344-9851>)

Repository CRAN

Date/Publication 2020-02-06 10:20:15 UTC
R topics documented:

as.Date.date_xx .......................... 2
as_yearqtr ................................ 4
c.date_xx ................................. 5
date_xx .................................... 6
date_xx_arithmetic ....................... 7
date_xx_arithmetic_disabled .............. 8
date_xx_breaks ............................ 9
date_xx_sequences ......................... 9
date_y ...................................... 10
date_ym .................................... 11
date_yq ..................................... 12
date_yw ..................................... 13
first_of_isoweek .......................... 14
first_of_isoyear ........................... 15
first_of_month ............................ 16
first_of_quarter ........................... 17
first_of_year ............................. 18
first_of_yq ................................ 19
format_date_xx ................... 20
format_ym ................................ 22
format_yq ................................ 23
format_yw ................................ 24
get_year .................................. 25
is_quarter_bounds ....................... 26
Ops.date_xx ............................... 27
print.date_xx ............................. 28
rep.date_xx ............................... 28
round.date_yq ............................ 29
scale_date_xx ............................ 30
Summary.date_xx ........................ 32
year.date_xx ............................. 33
yq ........................................... 34
%y+% ....................................... 35
[.date_xx ................................. 35
%y-%% ..................................... 36

Index 38

as.Date.date_xx  Coerce dint Objects to Base R Date Types

Description

All dint objects can be coerced to base R Date or Datetime (POSIXct) types. The resulting date will always default to the first possible Date/Datetime in this period.
as.Date$date_xx

Usage

## S3 method for class 'date_xx'
as.POSIXlt(x, tz = "", ...)

## S3 method for class 'date_xx'
as.POSIXct(x, tz = "", ...)

Sys.date_yq()
Sys.date_ym()
Sys.date_yw()

## S3 method for class 'date_y'
as.Date(x, ...)

## S3 method for class 'date_ym'
as.Date(x, ...)

## S3 method for class 'date_yq'
as.Date(x, ...)

## S3 method for class 'date_yw'
as.Date(x, ...)

Arguments

x       any R object

 tz       time zone specification to be used for the conversion, if one is required. System-specific (see time zones), but "" is the current time zone, and "GMT" is UTC (Universal Time, Coordinated). Invalid values are most commonly treated as UTC, on some platforms with a warning.

...       passed on to methods

Details

If lubridate is loaded, methods for lubridate generics (such as lubridate::month() and lubridate::year()) are also made available by dint.

Value

An Object of the appropriate base R type (Date, POSIXct, or POSIXlt)

Examples

as.Date(date_yq(2017, 2))
as.POSIXlt(date_yq(2017, 2))

# When coercing to datetime, the default timezone is UTC
as.POSIXct(date_yq(2017, 2))

---

**as_yearqtr**

Coerce to zoo yearqtr objects

**Description**

`as_yearqtr()` and `as_yearmon()` are included for interoperability with `zoo::yearqtr()`, an alternative year-quarter format that is based on a decimal representation as opposed to dint's integer representation of year-quarters. `as_yearweek()` follows a similar idea, but there is no corresponding S3 class in `zoo`. These functions were included for cases where you need a continuous representation of `date_xx` objects other than `base::Date()` (for example, they are used by `scale_date_xx`)

**Usage**

```r
as_yearqtr(x)
```

```r
# S3 method for class 'date_yq'
as_yearqtr(x)
```

```r
# S3 method for class 'yearqtr'
as_yearqtr(x)
```

```r
as_yearmon(x)
```

```r
# S3 method for class 'date_ym'
as_yearmon(x)
```

```r
# S3 method for class 'yearmon'
as_yearmon(x)
```

```r
as_yearweek(x)
```

```r
# S3 method for class 'date_yw'
as_yearweek(x)
```

```r
# S3 method for class 'yearweek'
as_yearweek(x)
```

**Arguments**

- `x` any R object

**Value**

a `zoo::yearqtr`, `zoo::yearmon` or `dint::yearweek` vector.
Examples

q <- date_yq(2016, 1:4)
as.numeric(q)
qzoo <- as_yearqtr(q)
as.numeric(qzoo)

m <- date_ym(2016, 1:12)
as.numeric(m)
mzoo <- as_yearmon(m)
as.numeric(mzoo)

w <- date_yw(2016, 1:52)
as.numeric(w)
wzoo <- as_yearweek(w)
as.numeric(wzoo)

c.date_xx

Concatenate date_xx Objects

Description

Concatenate date_xx Objects

Usage

## S3 method for class 'date_xx'
c(...)

Arguments

... date_yq, date_ym, date_yw or date_y vectors. All inputs must be of the same
type (or its unclassed integer equivalent) or faulty output is to be expected

Value

a vector of the same date_xx subclass as the first element of ...

Examples

c(date_yq(2000, 1:2), date_yq(2000, 3:3))

# raises an error
try(c(date_yq(2000, 1:2), date_ym(2000, 1:12)))
date_xx

A Superclass For All dint Objects

Description

Superclass for date_yq, date_ym, date_yw, and date_y.

make_date_xx can be used to create such objects when it is not know if month or quarter information is available.
is_date_xx() checks for date_xx objects.
date_xx() is an internally used constructor that should only be used by developers aspiring to extend the dint package.

Usage

date_xx(x, subclass)

make_date_xx(y, q = NULL, m = NULL)
is_date_xx(x)

Arguments

x Any R object
subclass subclass to assign
y, q, m Year, quarter, month. q and m are optional and at least one of them must be NULL.

Value

a date_xx Object, except for is_date_xx() which returns TRUE or FALSE
a date_xx Object for date_xx(), make_date_xx
is_date_xx() returns TRUE or FALSE depending on whether its argument is of type date_xx or not.

Examples

make_date_xx(2017)
make_date_xx(2017, 4)
x <- make_date_xx(2017, m = 4)
is_date_xx(x)
The arithmetic operations +, - as well as sequence generation with seq() are all supported for date_yq and date_ym objects. Other binary arithmetic operators are disabled (see date_xx_arithmetic_disabled).

## S3 method for class 'date_xx'

```r
x + y
```

## S3 method for class 'date_xx'

```r
x - y
```

### Arguments

- **x**: a date_yq or date_ym object
- **y**: an integer

### See Also

base::Arithmetic

### Examples

```r
q <- date_yq(2018, 1)

q + 5
q - 1
seq(q, q + 5)

m <- date_ym(2018, 12)
m + 1
m - 13
seq(m - 1, m + 1)
```
Date xx Arithmetic Disabled

Description

This page lists operators that are disabled for date_yq and date_ym objects.

Usage

```r
## S3 method for class 'date_xx'
x * y
## S3 method for class 'date_xx'
x / y
## S3 method for class 'date_xx'
x ^ y
## S3 method for class 'date_xx'
x %% y
## S3 method for class 'date_y'
x %/% y
## S3 method for class 'date_y'
x %% y
## S3 method for class 'date_y'
x %/% y
```

Arguments

- `x`: a date_yq or date_ym object
- `y`: an integer

See Also

date_xx_arithmetic, base::Arithmetic
**date_xx_breaks**

Pretty Breaks For date_xx Vectors

---

**Description**

(date_xa_breaks does not return breaks, but a function that calculates breaks. This is for compatibility with the breaks functions from scales such as scales::pretty_breaks(), and for ease of use with ggplot2.

**Usage**

```r
date_yq_breaks(n = 6)
date_ym_breaks(n = 6)
date_yw_breaks(n = 6)
```

**Arguments**

- `n` NULL or integer scalar. The desired maximum number of breaks. The breaks algorithm may choose less breaks if it sees fit.

**Value**

A function that calculates a maximum of `n` breaks for a date_xx vector

**Examples**

```r
x <- date_ym(2016, 1:12)
date_ym_breaks()(x)
date_ym_breaks(12)(x)
```

---

**date_xx_sequences**

---

**Description**

(date_xx Sequence Generation

```r
```
## A Simple S3-Class for Years

A simple data type for storing years. A `date_y` object is just an integer with an additional class attribute.

### Usage

```r
date_y(y)
```

```r
is_date_y(x)
```

```r
as_date_y(x)
```

### Arguments

- `y` year
- `x` any R object

### Description

A simple data type for storing years. A `date_y` object is just an integer with an additional class attribute.

### Examples

```r
# Create a date_y object
y <- date_y(2023)

is_date_y(y)  # TRUE
```

```r
# Convert a date object to a date_y object
as_date_y(y)
```

```r
# Create a sequence of years
seq(from = 2000, to = 2023)
```
**date_y**

**Value**

date_y returns an object of type date_y

is_date_y returns TRUE or FALSE depending on whether its argument is of type date_y or not.

as_date_y attempts to coerce its argument to date_y type

**See Also**

Other date_xx subclasses: date_y(), date_yq(), date_yw()

**Examples**

date_y(2013)

as_date_y(2016)

---

**Description**

A simple data type for storing year-month dates in a human readable integer format, e.g.: December 2012 is stored as 201212. Supports simple arithmetic operations such as + and - as well formatting.

**Usage**

date_y(y, m)

is_date_y(x)

as_date_y(x)

**Arguments**

<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
</tr>
<tr>
<td>m</td>
</tr>
<tr>
<td>x</td>
</tr>
</tbody>
</table>

**Value**

date_y returns an object of type date_y

is_date_y returns TRUE or FALSE depending on whether its argument is of type date_y or not.

as_date_y attempts to coerce its argument to date_y
Description

A simple data type for storing year-quarter dates in a human readable integer format, e.g.: 3.Quarter of 2012 is stored as 20123. Supports simple arithmetic operations such as + and - as well formatting.

Usage

date_yq(y, q)

is_date_yq(x)

as_date_yq(x)

Arguments

y year
q quarter (optional)
x any R object

Value

date_yq returns an object of type date_yq
is_date_yq returns TRUE or FALSE depending on whether its argument is of type date_yq or not.
as_date_yq attempts to coerce its argument to date_yq

See Also

format.date_yq(), seq.date_yq(), date_xxx_arithmetic()
Other date_xxx subclasses: date_ym(), date_yw(), date_y()
date_yw

Examples

date_yq(2013, 3)
as_date_yq(20161)

date_yw  A Simple S3-Class for Year-Isoweek Dates

Description
A simple data type for storing year-isoweek dates in a human readable integer format, e.g.: the 52nd
isoweek of 2012 is stored as 201252. Supports simple arithmetic operations such as + and – as well
formatting.

Usage

date_yw(y, w)
is_date_yw(x)
as_date_yw(x)

Arguments

y year
w week (optional)
x any R object

Value

date_yw returns an object of type date_yw
is_date_yw returns TRUE or FALSE depending on whether its argument is of type date_yw or not.
as_date_yw attempts to coerce its argument to date_yw

See Also

format.date_yw(), seq.date_yw(), date_xx_arithmetic()
Other date_xx subclasses: date_ym(), date_yq(), date_y()

Examples

date_yw(2013, 12)
as_date_yw(201612)
第一周的首日

### Description


### Usage

```r
first_of_isoweek(x)
```

```r
## Default S3 method:
first_of_isoweek(x)
```

```r
last_of_isoweek(x)
```

```r
## Default S3 method:
last_of_isoweek(x)
```

```r
first_of_yw(x, w = NULL)
```

```r
last_of_yw(x, w = NULL)
```

### Arguments

- **x** Anything that can be coerced to a date with `base::as.Date()`
- **w**  Two integer (vectors). `w` is optional and the interpretation of `x` will depend on whether `w` is supplied or not:
  - if only `x` is supplied, `x` will be passed to `as_date_yw()` (e.g. `x = 201604` means 4th isoweek of 2016)
  - if `x` and `w` are supplied, `x` is interpreted as year and `w` as week.

### Value

A Date

### See Also

`first_of_isoweek()`

### Examples

```r
first_of_isoweek("2016-06-04")
last_of_isoweek("2016-06-04")
first_of_yw(2016)
first_of_yw(2016)
```
first_of_isoyear

Get First / Last Day of the First and Last Isoweek of a Year

Description

Get First / Last Day of the First and Last Isoweek of a Year

Usage

first_of_isoyear(x)

## Default S3 method:
first_of_isoyear(x)

## S3 method for class 'date_yw'
first_of_isoyear(x)

## S3 method for class 'integer'
first_of_isoyear(x)

## S3 method for class 'numeric'
first_of_isoyear(x)

last_of_isoyear(x)

## Default S3 method:
last_of_isoyear(x)

## S3 method for class 'date_yw'
last_of_isoyear(x)

## S3 method for class 'integer'
last_of_isoyear(x)

## S3 method for class 'numeric'
last_of_isoyear(x)

Arguments

x anything that can be coerced to a Date
first_of_month  

Get First / Last Day of a Month

Description

Get First / Last Day of a Month
Get First or Last Day of Month From Year and Month

Usage

first_of_month(x)

## Default S3 method:
first_of_month(x)

class(x) = c("xy", "date")

last_of_month(x)

## Default S3 method:
last_of_month(x)

first_of_ym(x, m = NULL)

last_of_ym(x, m = NULL)

Arguments

x  
Anything that can be coerced to a date with base::as.Date()

m  
Two integer (vectors). m is optional and the interpretation of x will depend on whether m is supplied or not:

• if only x is supplied, x will be passed to as_date_ym() (e.g. x = 201604 means April 2016)
• if x and m are supplied, x is interpreted as year and m as month.

Value

a Date

See Also

first_of_month()

Examples

first_of_month("2016-06-04")
last_of_month("2016-06-04")
first_of_quarter

Description

Get First / Last Day of a Quarter

Usage

first_of_quarter(x)

## Default S3 method:
first_of_quarter(x)

last_of_quarter(x)

## Default S3 method:
last_of_quarter(x)

Arguments

x Anything that can be coerced to a date with \texttt{base::as.Date()}

Value

a \texttt{Date}

Examples

first_of_quarter("2016-06-04")
last_of_quarter("2016-06-04")
Get First / Last Day of a Year

Description

Get First / Last Day of a Year

Usage

`first_of_year(x)`

```r
## S3 method for class 'date_xx'
first_of_year(x)
```

```r
## S3 method for class 'integer'
first_of_year(x)
```

```r
## Default S3 method:
first_of_year(x)
```

```r
## S3 method for class 'numeric'
first_of_year(x)
```

```r
last_of_year(x)
```

```r
## S3 method for class 'date_xx'
last_of_year(x)
```

```r
## S3 method for class 'integer'
last_of_year(x)
```

```r
## Default S3 method:
last_of_year(x)
```

```r
## S3 method for class 'numeric'
last_of_year(x)
```

Arguments

- `x` Anything that can be coerced to a date with `base::as.Date()`

Value

- a Date


**first_of_yq**

---

**Examples**

```r
first_of_year("2016-06-04")
last_of_year("2016-06-04")
```

---

### Description

Get First or Last Day of Quarter From Year and Quarter

### Usage

```r
first_of_yq(x, q = NULL)
last_of_yq(x, q = NULL)
```

### Arguments

- **x**: Two integer (vectors). `q` is optional and the interpretation of `x` will depend on whether `q` is supplied or not:
  - if only `x` is supplied, `x` will be passed to `as_date_yq()` (e.g. `x = 20161` means first quarter of 2016)
  - if `x` and `q` are supplied, `x` is interpreted as year and `q` as quarter.
- **q**: Two integer (vectors). `q` is optional and the interpretation of `x` will depend on whether `q` is supplied or not:
  - if only `x` is supplied, `x` will be passed to `as_date_yq()` (e.g. `x = 20161` means first quarter of 2016)
  - if `x` and `q` are supplied, `x` is interpreted as year and `q` as quarter.

### Value

A Date

### See Also

`first_of_quarter()`

### Examples

```r
first_of_yq(2016, 1)
first_of_yq(20161)
```
Description

Format a date_xx

Usage

## S3 method for class 'date_y'
format(x, format = "%Y", ...)

## S3 method for class 'date_yq'
format(
  x,
  format = "%Y-Q%q",
  month_names = format(ISOdate(2000, 1:12, 1), "%B"),
  month_abb = format(ISOdate(2000, 1:12, 1), "%b"),
  ...
)

## S3 method for class 'date_ym'
format(
  x,
  format = "%Y-M%m",
  month_names = format(ISOdate(2000, 1:12, 1), "%B"),
  month_abb = format(ISOdate(2000, 1:12, 1), "%b"),
  ...
)

## S3 method for class 'date_yw'
format(x, format = "%Y-W%V", ...)

format_yq_iso(x)
format_yq_short(x)
format_yq_shorter(x)
format_ym_iso(x)
format_ym_short(x)
format_ym_shorter(x)
format_yw_iso(x)
format_yw_shorter(x)

Arguments

  x  any R object.
  format  A format that uses a subset of the same placeholders as \texttt{base::strptime()}: 

  \begin{itemize}
  \item \%Y Year with century (the full year)
  \item \%y Year without century (the last two digits of the year)
  \item \%m Month as a decimal numbers (01-12)
  \item \%B Full month name
  \item \%b Abbreviated month name
  \item \%V Week of the year as decimal number (01-53) as defined in ISO8601
  \end{itemize}

  Not all placeholders are supported for all \texttt{date_xx} subclasses. Literal "\%" can be escaped with "\%\%" (as in \texttt{base::sprintf()}).

  ... ignored
  month_names, month_abb  a character vector of length 12: Names and abbreviations for months that will be used for the placeholders "\%b" and "\%B". Defaults to the values for the current locale for compatibility with \texttt{base::strptime()}. 

Value

  a character vector

Formatting shorthands

Format shorthand functions in the form of \texttt{format_y*[preset]()} directly apply formatting presets to anything that can be coerced to a \texttt{date_xx}. This is notably handy as they can be used as a labeling function for \texttt{ggplot2} axes (see \texttt{vignette("dint")})

Examples

\begin{verbatim}
x <- date_ym(2018, c(1L, 10L, 3L, 6L, 4L, 5L, 7L, 12L, 2L, 9L, 8L, 11L))
fm <- "\%Y-\%m: \%B,\%b"

format(
x, 
  format = fm, 
  month_names = month.name, # built-in R constant for English names
  month_abb = month.abb
)
\end{verbatim}
**Description**

Coerce and Format to Year-Month Strings

**Usage**

```r
format_ym(x, m = NULL, format = "%Y-%M%m")
```

**Arguments**

- `x, m` Two integer (vectors). `m` is optional and the interpretation of `x` will depend on whether `m` is supplied or not:
  - if only `x` is supplied, `x` will be passed to `as_date_ym()` (e.g. `x = 201604` means April 2016)
  - if `x` and `m` are supplied, `x` is interpreted as year and `m` as month.

- `format` A format that uses a subset of the same placeholders as `base::strftime()`:

  - `%Y` Year with century (the full year)
  - `%y` Year without century (the last two digits of the year)
  - `%m` Month as a decimal numbers (01-12)
  - `%B` Full month name
  - `%b` Abbreviated month name
  - `%V` Week of the year as decimal number (01-53) as defined in ISO8601

Not all placeholders are supported for all `date_xx` subclasses. Literal "%" can be escaped with "\"" (as in `base::sprintf()`).

**Value**

a character vector

**Formatting shorthands**

Format shorthand functions in the form of `format_y*[preset]()` directly apply formatting presets to anything that can be coerced to a `date_xx`. This is notably handy as they can be used as a labeling function for `ggplot2` axes (see vignette("dint"))

**See Also**

`format.date_ym()`

Other coerce and format functions: `format_yq()`, `format_yw()`
format_yq

Examples

format_ym(2015, 5)
format_ym(201505, format = "short")
format_ym(201505, format = "shorter")

format_yq

Coerce and Format to Year-Quarter Strings

Description

Coerce and Format to Year-Quarter Strings

Usage

format_yq(x, q = NULL, format = "%Y-Q%q")

Arguments

x, q Two integer (vectors). q is optional and the interpretation of x will depend on whether q is supplied or not:
  • if only x is supplied, x will be passed to as_date_yq() (e.g. x = 20161 means first quarter of 2016)
  • if x and q are supplied, x is interpreted as year and q as quarter.

format A format that uses a subset of the same placeholders as base::strptime():

%Y Year with century (the full year)
%y Year without century (the last two digits of the year)
%m Month as a decimal numbers (01-12)
%B Full month name
%b Abbreviated month name
%V Week of the year as decimal number (01-53) as defined in ISO8601

Not all placeholders are supported for all date_xx subclasses. Literal "%" can be escaped with "%%" (as in base::sprintf())

Value

a character vector

Formatting shorthands

Format shorthand functions in the form of format_y*[preset]() directly apply formatting presets to anything that can be coerced to a date_xx. This is notably handy as they can be used as a labeling function for ggplot2 axes (see vignette("dint"))
Coerce and Format to Year-Isoweek Strings

Description

Coerce and Format to Year-Isoweek Strings

Usage

format_yw(x, w = NULL, format = "%Y-%W\%V")

Arguments

x, w
  Two integer (vectors). w is optional and the interpretation of x will depend on whether w is supplied or not:
  • if only x is supplied, x will be passed to as_date_yw() (e.g. x = 201604 means 4th isoweek of 2016)
  • if x and w are supplied, x is interpreted as year and w as week.

format
  A format that uses a subset of the same placeholders as base::strptime():
  • %Y Year with century (the full year)
  • %y Year without century (the last two digits of the year)
  • %m Month as a decimal numbers (01-12)
  • %B Full month name
  • %b Abbreviated month name
  • %V Week of the year as decimal number (01-53) as defined in ISO8601

Value

a character vector
get_year

Formatting shorthands

Format shorthand functions in the form of format_y*_{[preset]}() directly apply formatting presets to anything that can be coerced to a date_xx. This is notably handy as they can be used as a labeling function for ggplot2 axes (see vignette("dint"))

See Also

format.date_yw()

Other coerce and format functions: format_ym(), format_yq()

Examples

format_yw(2015, 5)
format_yw(201505, format = "%Y.%V")
format_yw(as_date_yw(201505), format = "%y.%V")

get_year Get Year, Quarter, Month or Isoweek

Description

Get Year, Quarter, Month or Isoweek

Usage

get_year(x)

get_quarter(x)

get_month(x)

get_isoweek(x)

get_isoyear(x)

Arguments

x a date_xx or any R object that can be coerced to POSIXlt

Details

If you use lubridate in addition to dint, you can also use lubridate::year(), lubridate::month() and lubridate::quarter() with dint objects.

Value

an integer vector.
is_quarter_bounds

See Also

lubridate::year(), lubridate::month(), lubridate::quarter()

Examples

```r
x <- date_yq(2016, 2)
gt_year(x)
## Not run:
library(lubridate)
year(x)

## End(Not run)

x <- date_yq(2016, 2)
gt_quarter(x)
## Not run:
library(lubridate)
quarter(x)

## End(Not run)

x <- date_yq(2016, 2)
gt_month(x)
## Not run:
library(lubridate)
month(x)

## End(Not run)

x <- date_yw(2016, 2)
gt_isoweek(x)

get_isoyear(as.Date("2018-01-01"))
get_isoyear(as.Date("2016-01-01"))
```

---

is_quarter_bounds  Useful Predicates for Dates

Description

is_first_of_quarter(), is_last_of_quarter(), is_first_of_year() and is_last_of_year() check whether a Date is the first or respectively the last day of a quarter/year. is_quarter_bounds() and is_year_bounds checks whether two Date vectors mark the bounds of (the same) quarters.

Usage

is_quarter_bounds(first, last)

is_first_of_quarter(x)
is_last_of_quarter(x)

is_year_bounds(first, last)

is_first_of_year(x)

is_last_of_year(x)

is_Date(x)

is_POSIXlt(x)

Arguments

x, first, last  Date vectors

Value

a logical vector

Examples

x <- as.Date(c("2018-01-01", "2018-03-31", "2018-02-14"))

is_first_of_year(x)

is_first_of_quarter(x)

is_last_of_quarter(x)

is_quarter_bounds(x[[1]], x[[2]])

is_quarter_bounds(x[[2]], x[[3]])

---

Description

Comparison Operators for date_xx

Usage

## S3 method for class 'date_xx'
Ops(e1, e2)

Arguments

e1, e2  Objects with the same date_xx subclass (one of them can also be integer)

Value

a logical scalar
Examples

date_yq(2015, 1) < date_yq(2015, 2)

# comparison with integers is ok
date_yq(2015, 1) < 20152

# but two different date_xx cannot be compared
try(date_yq(2015, 1) < date_ym(2015, 2))

print.date_xx  
Print a date_xx Object

Description
Print a date_xx Object

Usage

## S3 method for class 'date_xx'
print(x, ...)

Arguments

x  A date_xx object
...

Value

x (invisibly)

rep.date_xx  
Replicate Elements of date_xx Vectors

Description
Replicate Elements of date_xx Vectors

Usage

## S3 method for class 'date_xx'
rep(x, ...)

Arguments

x  a date_xx
...

Value

base::rep()
round.date_yq

Value

a vector of the same date.xx subclass as x

---

round.date_yq Rounding of date.xx

Description

Rounds a date.xx to the first unit of the current year, or the first unit of the next year.

Usage

```r
## S3 method for class 'date_yq'
round(x, digits = NULL)
## S3 method for class 'date_ym'
round(x, digits = NULL)
## S3 method for class 'date_yw'
round(x, digits = NULL)
## S3 method for class 'date_xx'
ceiling(x)
## S3 method for class 'date_xx'
floor(x)
```

Arguments

- `x` any date.xx object
- `digits` ignored, only there for compatibility with `base::round()`

Value

a date.xx of the same subclass as x

Examples

```r
round(date_yq(2018, 2))
round(date_yq(2018, 3))
round(date_ym(2018, 6))
round(date_ym(2018, 7))
round(date_yw(2018, 26))
round(date_yw(2018, 27))
```
Description

The scale_*_date_** functions provide nice defaults for plotting the appropriate date_xx subclass, but come with a limited number of configuration options. If you require more finetuning, you can convert date_xx vectors with as.Date() and use ggplot2::scale_x_date().

Usage

```r
scale_x_date_yq(
  name = "Quarter",
  breaks = date_yq_breaks(),
  labels = ggplot2::waiver(),
  limits = NULL,
  position = "bottom"
)

scale_y_date_yq(
  name = "Quarter",
  breaks = date_yq_breaks(),
  labels = ggplot2::waiver(),
  limits = NULL,
  position = "left"
)

scale_x_date_ym(
  name = "Month",
  breaks = date_ym_breaks(),
  labels = ggplot2::waiver(),
  limits = NULL,
  position = "bottom"
)

scale_y_date_ym(
  name = "Month",
  breaks = date_ym_breaks(),
  labels = ggplot2::waiver(),
  limits = NULL,
  position = "left"
)

scale_x_date_yw(
  name = "Week",
  breaks = date_yw_breaks(),
  labels = ggplot2::waiver(),
  limits = NULL,
  position = "bottom"
)
```

```r
```
limits = NULL,
position = "bottom"
)

scale_y_date_yw(
  name = "Week",
  breaks = date_yw_breaks(),
  labels = ggplot2::waiver(),
  limits = NULL,
  position = "left"
)

### Arguments

- **name**
  The name of the scale. Used as the axis or legend title. If `waiver()`, the default, the name of the scale is taken from the first mapping used for that aesthetic. If `NULL`, the legend title will be omitted.

- **breaks**
  One of:
  - `NULL` for no breaks
  - `ggplot2::waiver()` for automatic breaks (see `date_xx_breaks()`)
  - A `date_xx` vector of breaks
  - A function that takes the limits as input and returns breaks as output

- **labels**
  One of:
  - `NULL` for no labels
  - `ggplot2::waiver()` for the default labels computed by the transformation object
  - A character vector giving labels (must be same length as breaks, so it’s a good idea to specify manual breaks if you use manual labels)
  - A function that takes the breaks as input and returns labels as output

- **limits**
  One of:
  - `NULL` to use the default scale range
  - A numeric vector of length two providing limits of the scale. Use `NA` to refer to the existing minimum or maximum
  - A function that accepts the existing (automatic) limits and returns new limits
  Note that setting limits on positional scales will **remove** data outside of the limits. If the purpose is to zoom, use the limit argument in the coordinate system (see `coord_cartesian()`).

- **position**
  For position scales, The position of the axis. left or right for y axes, top or bottom for x axes.

### Examples

```r
if (require("ggplot2", quietly = TRUE)){
  dd <- data.frame(date = seq(date_yq(2016, 1), date_yq(2018, 1)), V1 = 1:9)
  p <- ggplot(dd, aes(x = date, y = V1)) +
  ```
```r
# automatically uses the proper scale
p + scale_x_date_yq("quarters with default spacing")
p + scale_x_date_yq(breaks = date_yq_breaks(3))

# Different ways to specify breaks and labels
p <- ggplot(
  data.frame(date = seq(date_yq(2012, 4), date_yq(2018, 4)), V1 = 1:25),
  aes(x = date, y = V1)
) +
  geom_point()

p + scale_x_date_yq(labels = waiver()) + ggtitle("auto Labels")
p + scale_x_date_yq(labels = NULL) + ggtitle("no Labels")
p + scale_x_date_yq(labels = LETTERS[1:4]) + ggtitle("manual Labels")
p + scale_x_date_yq(labels = format_yq_iso) + ggtitle("function Labels")

p + scale_x_date_yq(breaks = waiver()) + ggtitle("auto breaks")
p + scale_x_date_yq(breaks = NULL) + ggtitle("no breaks")
p + scale_x_date_yq(breaks = date_yq(2013, 2:3)) + ggtitle("manual breaks")
p + scale_x_date_yq(breaks = date_yq_breaks(1)) + ggtitle("function breaks")
```

---

### Summary.date_xx

**Maxima and Minima for date_xx**

**Description**

Maxima and Minima for date_xx

**Usage**

```r
## S3 method for class 'date_xx'
Summary(...., na.rm)
```

**Arguments**

- `...` date_xx vectors with the same subclass
- `na.rm` logical: should missing values be removed?

**Value**

for `min()` and `max()` a scalar of the same date_xx subclass as it’s input, for range a vector of length 2
**year.date_xx**

### Examples

```r
min(date_yq(2014, 1), date_yq(2014, 2))

# raises an error
try(min(date_yq(2014, 1), date_ym(2014, 2)))
```

---

**Get Year, Quarter or Month (lubridate Compatibility)**

### Description

See [lubridate::year()](https://github.com/tidyverse/lubridate) and [lubridate::month()](https://github.com/tidyverse/lubridate)

### Usage

```r
year.date_xx(x)
```

```r
month.date_xx(x, label = FALSE, abbr = TRUE, locale = Sys.getlocale("LC_TIME"))
```

```r
isoweek.date_xx(x)
```

### Arguments

- **x**: a `date_xx` or any R object that can be coerced to `POSIXlt`
- **label**: logical. TRUE will display the month as a character string such as "January." FALSE will display the month as a number.
- **abbr**: logical. FALSE will display the month as a character string label, such as "January". TRUE will display an abbreviated version of the label, such as "Jan". abbr is disregarded if label = FALSE.
- **locale**: for month, locale to use for month names. Default to current locale.

### See Also

[get_year](https://github.com/tidyverse/lubridate)

### Examples

```r
## Not run:
library(lubridate)
month(x)
month(x, label = TRUE)
```

```r
## End(Not run)
```

```r
## Not run:
library(lubridate)
```
yq isoweek(x)

## End(Not run)

---

**yq**  
*Parse Dates With Year and Quarter Components*

**Description**

These are generic parsers for year/quarter/month formats that work with nearly all possible year/quarter formats. The only prerequisite is that `x` contains a 4-digit-year and a 1-digit-quarter or 2-digit-month and no additional numbers.

**Usage**

```r
yq(x, quiet = FALSE)
qy(x, quiet = FALSE)
ym(x, quiet = FALSE)
my(x, quiet = FALSE)
```

**Arguments**

- `x`: a character vector
- `quiet`: a logical scalar. If TRUE warnings on parsing failures are suppressed.

**Value**

a `date_yq` or `date_ym` vector

**Examples**

```r
yq("2018 1")
qy("1st Quarter 2019")
```

#' # Works even for filenames, as long as they contain no additional numbers

```r
yq("business_report-2018_1.pdf")
my("business_report-082018.pdf")
```
Description

Works exactly like subsetting base vectors via [], but preserves the date_xx class and subclasses. The replacement functions [<- and [[<- conduct additional checks before assignment to prevent the generation of degenerate date_xx vectors (see examples).

Usage

```r
## S3 method for class 'date_xx'
x[i]

## S3 replacement method for class 'date_yq'
x[i] <- value

## S3 replacement method for class 'date_ym'
x[i] <- value

## S3 replacement method for class 'date_yw'
x[i] <- value

## S3 method for class 'date_xx'
x[[i]]

## S3 replacement method for class 'date_yq'
x[[i]] <- value

## S3 replacement method for class 'date_ym'
x[[i]] <- value

## S3 replacement method for class 'date_yw'
x[[i]] <- value
```

Arguments

- `x`: object from which to extract element(s) or in which to replace element(s).
- `i`: indices specifying elements to extract or replace. Indices are numeric or character vectors or empty (missing) or NULL. Numeric values are coerced to integer as by `as.integer` (and hence truncated towards zero). Character vectors will be matched to the names of the object (or for matrices/arrays, the dimnames): see ‘Character indices’ below for further details.

For [[-indexing only: i, j, ... can be logical vectors, indicating elements/slices to select. Such vectors are recycled if necessary to match the corresponding extent. i, j, ... can also be negative integers, indicating elements/slices to leave out of the selection.
When indexing arrays by [ a single argument i can be a matrix with as many columns as there are dimensions of x; the result is then a vector with elements corresponding to the sets of indices in each row of i.

An index value of NULL is treated as if it were integer(0).

Value

A vector of the same class as x or a vector of integers that correspond to the internal representation date_yq/date_ym/date_yw objects (see examples)

See Also

base::Extract

Examples

```r
x <- date_yq(2016, 1:4)
x[[2]]
x[1] <- date_yq(2016, 3)
x[2] <- 20164  # 2016, 4th quarter
x[1:2]

# Trying to assign illegal values for the respective date_xx type raises an error
try(x[2] <- 20165)

x <- date_ym(2016, 1:3)
x[1] <- 201610  # October 2016

x <- date_yw(2016, 50:52)
x[1] <- 201649  # 2016, week 52
```

### Add/Subtract Year

Description

Add/Subtract Year

Usage

```r
x %y+% y
x %y-% y
```

## S3 method for class 'date_y'

## S3 method for class 'date_y'

```r
x %y+% y

## S3 method for class 'date_yq'

x %y-% y

## S3 method for class 'date_yq'

x %y+% y

## S3 method for class 'date_ym'

x %y-% y

## S3 method for class 'date_ym'

x %y+% y

## S3 method for class 'date_yw'

x %y-% y

## S3 method for class 'date_yw'

x %y+% y
```

### Arguments

- `x` a `date_xx` vector
- `y` an integer vector of years

### Examples

```r
date_yq(2017, 1) %y+% 1
date_yq(2017, 1) %y-% 1
date_ym(2017, 1) %y+% 1
date_ym(2017, 1) %y-% 1
```
Index

*.date_xx
  (date_xx_arithmetic_disabled), 8
+ .date_xx (date_xx_arithmetic), 7
- .date_xx (date_xx_arithmetic), 7
/.date_xx
  (date_xx_arithmetic_disabled), 8
[.date_xx, 35
[[- .date_ym ([.date_xx), 35
[[- .date_yq ([.date_xx), 35
[[- .date_yw ([.date_xx), 35
[[<-.date_xx ([.date_xx), 35
[[<-.date_yq ([.date_xx), 35
[[<-.date_yw ([.date_xx), 35
%/% .date_xx
  (date_xx_arithmetic_disabled), 8
%/% .date_y
  (date_xx_arithmetic_disabled), 8
%% .date_xx
  (date_xx_arithmetic_disabled), 8
%% .date_y
  (date_xx_arithmetic_disabled), 8
%y~%(%y~%), 36
%y%+, 36
^ .date_xx
  (date_xx_arithmetic_disabled), 8
as.Date(), 30
as.Date.date_xx, 2
as.Date.date_y (as.Date.date_xx), 2
as.Date.date_ym (as.Date.date_xx), 2
as.Date.date_yq (as.Date.date_xx), 2
as.Date.date_yw (as.Date.date_xx), 2
as.integer, 35
as.POSIXct.date_xx (as.Date.date_xx), 2
as.POSIXlt.date_xx (as.Date.date_xx), 2
as_date_y (date_y), 10
as_date_ym (date_ym), 11
as_date_ym(), 16–18
as_date_yq (date_yq), 12
as_date_yq(), 19–23
as_date_yw (date_yw), 13
as_date_yw(), 14–24
as_yearmon (as_yearqtr), 4
as_yearqtr, 4
as_yearweek (as_yearqtr), 4
base::Arithmetic, 7, 8
base::as.Date(), 14, 16–18
base::Date(), 4
base::Extract, 36
base::rep(), 28
base::round(), 29
base::sprintf(), 21–24
base::strptime(), 21–24
c.date_xx, 5
ceil.date_xx (round.date_yq), 29
go_cartesian()., 31
Date, 14, 16–19
date_xx, 6, 10, 25, 28, 30, 33, 37
date_xx_arithmetic, 7, 8
date_xx_arithmetic(), 12, 13
date_xx_arithmetic_disabled, 7, 8
date_xx_breaks, 9
date_xx_breaks(), 31
date_xx_sequences, 9
date_y, 6, 10, 12, 13
date_ym, 6–8, 10, 11, 12, 13
date_ym_breaks (date_xx_breaks), 9
date_yq, 6–8, 10–12, 12, 13
date_yq_breaks (date_xx_breaks), 9
INDEX

date_yw, 6, 11, 12, 13
date_yw_breaks (date_xx_breaks), 9
dimnames, 35

first_of_isoweek, 14
first_of_isoweek(), 14
first_of_isoyear, 15
first_of_month, 16
first_of_month(), 16
first_of_quarter, 17
first_of_quarter(), 19
first_of_year, 18
first_of_ym (first_of_month), 16
first_of_ym(), 14
first_of_yq, 19
first_of_yq(), 14
first_of_yw (first_of_isoweek), 14
floor.date_xx (round.date_yq), 29
format.date_y (format_date_xx), 20
format.date_ym (format_date_xx), 20
format.date_ym(), 12, 22, 28
format.date_yq (format_date_xx), 20
format.date_yq(), 12, 24, 28
format.date_yw (format_date_xx), 20
format.date_yw(), 13, 25
format_date_xx, 20
format_ym, 22, 24, 25
format_ym_iso (format_date_xx), 20
format_ym_short (format_date_xx), 20
format_ym_shorter (format_date_xx), 20
format_yq, 22, 23, 25
format_yq_iso (format_date_xx), 20
format_yq_short (format_date_xx), 20
format_yq_shorter (format_date_xx), 20
format_yw, 22, 24, 24
format_yw_iso (format_date_xx), 20
format_yw_short (format_date_xx), 20
format_yw_shorter (format_date_xx), 20

get_isoweek (get_year), 25
get_isoyear (get_year), 25
get_month (get_year), 25
get_quarter (get_year), 25
get_year, 25, 33

ggplot2::scale_x_date(), 30

is_Date (is_quarter_bounds), 26
is_date_xx (date_xx), 6
is_date_y (date_y), 10

is_date_ym (date_ym), 11
is_date_yq (date_yq), 12
is_date_yw (date_yw), 13
is_first_of_quarter
(is_quarter_bounds), 26
is_first_of_year (is_quarter_bounds), 26
is_last_of_quarter (is_quarter_bounds), 26
is_last_of_year (is_quarter_bounds), 26
is_POSIXlt (is_quarter_bounds), 26
is_quarter_bounds, 26
is_year_bounds (is_quarter_bounds), 26
isweek.date_xx (year.date_xx), 33

last_of_isoweek (first_of_isoweek), 14
last_of_isoyear (first_of_isoyear), 15
last_of_month (first_of_month), 16
last_of_quarter (first_of_quarter), 17
last_of_year (first_of_year), 18
last_of_ym (first_of_month), 16
last_of_yq (first_of_yq), 19
last_of_yw (first_of_isoweek), 14
lubridate::month(), 3, 25, 26, 33
lubridate::quarter(), 25, 26
lubridate::year(), 3, 25, 26, 33

make_date_xx (date_xx), 6
month (year.date_xx), 33
my (yq), 34

names, 35

Ops.date_xx, 27

print.date_xx, 28

qy (yq), 34

rep.date_xx, 28
round.date_ym (round.date_yq), 29
round.date_yq, 29
round.date_yw (round.date_yq), 29

scale_date_xx, 4, 30
scale_x_date_ym (scale_date_xx), 30
scale_x_date_yq (scale_date_xx), 30
scale_x_date_yw (scale_date_xx), 30
scale_y_date_ym (scale_date_xx), 30
scale_y_date_yq (scale_date_xx), 30
scale_y_date_yw (scale_date_xx), 30
scales::pretty_breaks(), 9
seq.date_ym(date_xx_sequences), 9
seq.date_ym(), 12
seq.date_yq(date_xx_sequences), 9
seq.date_yq(), 12
seq.date_yw(date_xx_sequences), 9
seq.date_yw(), 13
Summary.date_xx, 32
Sys.date_ym(as.Date.date_xx), 2
Sys.date_yq(as.Date.date_xx), 2
Sys.date_yw(as.Date.date_xx), 2

time zones, 3

year(year.date_xx), 33
year.date_xx, 33
ym(yq), 34
yq, 34

zoo::yearmon, 4
zoo::yearqtr, 4
zoo::yearqtr(), 4