Package 'dint'

November 20, 2024

Type Package

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

Version 2.1.5

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

URL https://github.com/s-fleck/dint

BugReports https://github.com/s-fleck/dint/issues

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

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.0.2

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 2024-11-20 15:20:02 UTC

Contents

`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.

Usage

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

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

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	a character string. The 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 then `base::Date()` (for example, they are used by scale_date_xx)

Usage

```
as_yearqtr(x)

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

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

as_yearmon(x)

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

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

as_yearweek(x)

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

## 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)
```

`date_xx_arithmetic` *date_xx Arithmetic Operations*

Description

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).

Usage

```
## S3 method for class 'date_xx'
x + y

## S3 method for class 'date_xx'
x - y
```

Arguments

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

See Also

base::Arithmetic

Examples

```
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` *Arithmetic Operations Disabled for date_xx*

Description

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

Usage

```
## 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_xx'
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_*_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

```
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

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

date_xx_sequences *date_xx Sequence Generation*

Description

date_xx Sequence Generation

Usage

```
## S3 method for class 'date_yw'
seq(from, to, by = 1L, ...)

## S3 method for class 'date_yq'
seq(from, to, by = 1L, ...)

## S3 method for class 'date_ym'
seq(from, to, by = 1L, ...)
```

Arguments

from, to	the starting and (maximal) end value of the sequence. Must be of the same class (i.e. both must be a date_yq, date_ym, etc..)
by	a positive integer scalar to increment the sequence with (either in quarters, months or isoweeks, depending on the class of from/to)
...	ignored

Value

an integer vector with the same date_xx subclass as from/to

---

date_y *A Simple S3-Class for Years*

Description

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

Usage

```
date_y(y)

is_date_y(x)

as_date_y(x)
```

Arguments

y	year
x	any R object

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_m` attempts to coerce its argument to `date_y` type

See Also

Other date_xx subclasses: `date_ym()`, `date_yq()`, `date_yw()`

Examples

```
date_y(2013)

as_date_y(2016)
```

---

date_ym *A Simple S3-Class for Year-Month Dates*

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_ym(y, m)

is_date_ym(x)

as_date_ym(x)
```

Arguments

y	year
m	month (optional)
x	any R object

Value

`date_ym` returns an object of type `date_ym`

`is_date_ym` returns `TRUE` or `FALSE` depending on whether its argument is of type `date_ym` or not.

`as_date_ym` attempts to coerce its argument to `date_ym`

See Also

format.date_ym(), seq.date_ym(), date_xx_arithmetic()

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

Examples

```
date_ym(2013, 12)

as_date_ym(201612)
```

---

date_yq *A Simple S3-Class for Year-Quarter Dates*

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_xx_arithmetic()

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

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)
```

`first_of_isoweek` *Get First / Last Day of an Isoweek*

Description

`first_of_yw()` is equivalent with `first_of_isoweek()` and only included for symmetry with `first_of_yq()` and `first_of_ym()`.

Usage

```
first_of_isoweek(x)

## Default S3 method:
first_of_isoweek(x)

last_of_isoweek(x)

## Default S3 method:
last_of_isoweek(x)

first_of_yw(x, w = NULL)

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

```
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)

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_ym(2016, 1)
first_of_ym(201601)
```

`first_of_quarter` *Get First / Last Day of a 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 `base::as.Date()`

Value

a Date

Examples

```
first_of_quarter("2016-06-04")
last_of_quarter("2016-06-04")
```

first_of_year *Get First / Last Day of a Year*

Description

Get First / Last Day of a Year

Usage

```
first_of_year(x)

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

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

## Default S3 method:
first_of_year(x)

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

last_of_year(x)

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

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

## Default S3 method:
last_of_year(x)

## 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

Examples

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

`first_of_yq` *Get First or Last Day of Quarter From Year and Quarter*

Description

Get First or Last Day of Quarter From Year and Quarter

Usage

```
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

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

format_date_xx *Format a date_xx*

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_short(x)

format_yw_shorter(x)
```

Arguments

x	any R object.
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()`).

...	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 `base::strptime()`.

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")`)

Examples

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

format(
  x,
  format = fm,
  month_names = month.name,  # built-in R constant for English names
  month_abb = month.abb
)
```

format_ym *Coerce and Format to Year-Month Strings*

Description

Coerce and Format to Year-Month Strings

Usage

```
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::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"))

See Also

format.date_ym()

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

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")`)

See Also

format.date_yq()

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

Examples

```
format_yq(2015, 1)
format_yq(20151, format = "short")
format_yq(20151, format = "shorter")
```

---

format_yw *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 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_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.

See Also

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

Examples

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

## End(Not run)

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

## End(Not run)

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

## End(Not run)
x <- date_yw(2016, 2)
get_isoweek(x)

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

---

increment *Increment date_xx objects*

Description

Increment date_xx objects

Usage

```
increment(x, inc = 1)

## S3 method for class 'date_yq'
increment(x, inc)

## S3 method for class 'date_ym'
```

```
increment(x, inc)

## S3 method for class 'date_yw'
increment(x, inc)

## S3 method for class 'date_y'
increment(x, inc)
```

Arguments

x	object to increment
inc	Value by which to increment (usually integer)

Value

An object of the same type as x increment by inc

---

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]])
```

`Ops.date_xx` *Comparison Operators for date_xx*

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
...	passed on to `format.date_yq()` or `format.date_ym()`

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
...	passed on to `base::rep()`

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

```
## 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

```
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))
```

scale_date_xx *ggplot2 Scales For date_xx Objects*

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

```
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"
)

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. Also accepts rlang lambda function notation. 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

```
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)) +
  geom_point()

p  # 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

```
## 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

Examples

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

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

`year.date_xx` *Get Year, Quarter or Month (lubridate Compatibility)*

Description

See `lubridate::year()` and `lubridate::month()`

Usage

```
year.date_xx(x)

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

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

Examples

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

## End(Not run)

## Not run:
  library(lubridate)
  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

```
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

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

#' # Works even for filenames, as long as they contain no additional numbers
yq("business_report-2018_1.pdf")
my("business_report-082018.pdf")
```

[.date_xx *Extract or Replace Elements of a date_xx*

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

```
## 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 or whole numbers as by as.integer or for large values by trunc (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)

Value

a date_xx vector

See Also

base::Extract

Examples

```
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
```

%y+% *Add/Subtract Year*

Description

Add/Subtract Year

Usage

```
x %y+% y

x %y-% y

## S3 method for class 'date_y'
x %y+% y

## S3 method for class 'date_y'
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

```
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