

Package ‘grates’

November 20, 2024

Title Grouped Date Classes

Version 1.3.0

Description Provides a coherent interface and implementation for creating grouped date classes. This package is part of the RECON (<<https://www.repidemicsconsortium.org/>>) toolkit for outbreak analysis.

URL <https://www.reconverse.org/grates/>,
<https://github.com/reconverse/grates>

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Depends R (>= 3.6.0)

Suggests knitr, ggplot2, scales, vctrs, rlang, markdown, outbreaks, testthat (>= 3.0.0),

VignetteBuilder knitr

Config/testthat/edition 3

Imports utils

NeedsCompilation no

Author Tim Taylor [aut, cre] (<<https://orcid.org/0000-0002-8587-7113>>)

Maintainer Tim Taylor <tim.taylor@hiddenelephants.co.uk>

Repository CRAN

Date/Publication 2024-11-20 10:00:02 UTC

Contents

as_epiweek	2
as_isoweek	4
as_month	5
as_period	6
as_year	8
as_yearmonth	9

as_yearquarter	10
as_yearweek	11
boundaries	13
epiweek	14
grouped_date_accessors	15
isoweek	17
new_epiweek	18
new_isoweek	19
new_month	20
new_period	21
new_yearmonth	22
new_yearquarter	23
new_yearweek	23
print.grates_int_period	24
print.grates_month	25
print.grates_period	26
print.grates_year	26
print.grates_yearmonth	27
print.grates_yearquarter	27
scale_x_grates_epiweek	28
scale_x_grates_isoweek	28
scale_x_grates_month	29
scale_x_grates_period	30
scale_x_grates_year	31
scale_x_grates_yearmonth	32
scale_x_grates_yearquarter	33
scale_x_grates_yearweek	33
year	36
yearmonth	36
yearquarter	37
yearweek	38

Index 40

as_epiweek	<i>Coerce to a epiweek object</i>
------------	-----------------------------------

Description

Generic for conversion to <grates_epiweek>

Usage

```
as_epiweek(x, ...)
```

```
## Default S3 method:
as_epiweek(x, ...)
```

```
## S3 method for class 'Date'
as_epiweek(x, ...)

## S3 method for class 'POSIXt'
as_epiweek(x, ...)

## S3 method for class 'character'
as_epiweek(x, format, tryFormats = c("%Y-%m-%d", "%Y/%m/%d"), ...)

## S3 method for class 'factor'
as_epiweek(x, format, tryFormats = c("%Y-%m-%d", "%Y/%m/%d"), ...)
```

Arguments

x	R object.
...	Other values passed to as.Date().
format	[character] Passed to as.Date() unless format = "yearweek" in which case input is assumed to be in the form "YYYY-Wxx". If not specified, it will try tryFormats one by one on the first non-NA element, and give an error if none works. Otherwise, the processing is via strptime() whose help page describes available conversion specifications.
tryFormats	[character] Format strings to try if format is not specified.

Details

- Date, POSIXct, and POSIXlt are converted with the timezone respected.
- Character objects are first coerced to date via as.Date() unless format = "yearweek" in which case input is assumed to be in the form "YYYY-Wxx" and parsed accordingly.

Value

A <grates_epiweek> object.

See Also

new_epiweek() and as.Date().

Examples

```
as_epiweek(Sys.Date())
as_epiweek(as.POSIXct("2019-03-04 01:01:01", tz = "America/New_York"))
as_epiweek("2019-05-03")
as_epiweek("2019-W12", format = "yearweek")
```

as_isoweek

Coerce to a isoweek object

Description

Generic for conversion to <grates_isoweek>

Usage

```
as_isoweek(x, ...)

## Default S3 method:
as_isoweek(x, ...)

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

## S3 method for class 'POSIXt'
as_isoweek(x, ...)

## S3 method for class 'character'
as_isoweek(x, format, tryFormats = c("%Y-%m-%d", "%Y/%m/%d"), ...)

## S3 method for class 'factor'
as_isoweek(x, format, tryFormats = c("%Y-%m-%d", "%Y/%m/%d"), ...)
```

Arguments

x	R object.
...	Other values passed to as.Date().
format	[character] Passed to as.Date() unless format = "yearweek" in which case input is assumed to be in the form "YYYY-Wxx". If not specified, it will try tryFormats one by one on the first non-NA element, and give an error if none works. Otherwise, the processing is via strptime() whose help page describes available conversion specifications.
tryFormats	[character] Format strings to try if format is not specified.

Details

- Date, POSIXct, and POSIXlt are converted with the timezone respected.
- Character objects are first coerced to date via as.Date() unless format = "yearweek" in which case input is assumed to be in the form "YYYY-Wxx" and parsed accordingly.

Value

A <grates_isoweek> object.

See Also

new_isoweek() and as.Date().

Examples

```
as_isoweek(Sys.Date())
as_isoweek(as.POSIXct("2019-03-04 01:01:01", tz = "America/New_York"))
as_isoweek("2019-05-03")
as_isoweek("2019-W12", format = "yearweek")
```

as_month

Coerce an object to month

Description

as_month() is a generic for coercing input in to <grates_month>.

Usage

```
as_month(x, n, ...)
```

Default S3 method:
as_month(x, n, ...)

S3 method for class 'Date'
as_month(x, n, ...)

S3 method for class 'POSIXt'
as_month(x, n, ...)

S3 method for class 'character'
as_month(x, n, ...)

S3 method for class 'factor'
as_month(x, n, ...)

Arguments

x An R object.
Character input is first parsed using as.Date().
POSIXt inputs are converted with the timezone respected.

n [integer]
 Number of months that are being grouped. Must be greater than 1 (use as_yearmonth() for this case).

... Only used For character input where additional arguments are passed through to as.Date().

Value

A <grates_month> object.

Note

Internally grates_month objects are stored as the position, starting at 0, of n-month groups since the Unix Epoch (1970-01-01). Here n-months is taken to mean a 'grouping of n consecutive months'. Precision is only to the month level (i.e. the day of the month is always dropped).

References

The algorithm to convert between dates and months relative to the UNIX Epoch comes from the work of Davis Vaughan in the unreleased [datea](#) package.

See Also

as.Date()

Examples

```
as_month("2019-05-03", n = 4L)
as_month(as.POSIXct("2019-03-04 01:01:01", tz = "America/New_York"), n = 2)
```

as_period

Coerce an object to period

Description

as_period() is a generic for coercing input in to <grates_period>.

Usage

```
as_period(x, n, ...)

## Default S3 method:
as_period(x, n = 1L, offset = 0L, ...)

## S3 method for class 'Date'
as_period(x, n = 1L, offset = 0L, ...)
```

```
## S3 method for class 'POSIXt'
as_period(x, n = 1L, offset = 0L, ...)

## S3 method for class 'character'
as_period(x, n = 1L, offset = 0L, ...)

## S3 method for class 'factor'
as_period(x, n = 1L, offset = 0L, ...)
```

Arguments

x	An R object: <ul style="list-style-type: none"> • Character input is first parsed using <code>as.Date()</code>. • POSIXt inputs are converted with the timezone respected.
n	[integer] Number of days that are being grouped.
...	Only used for character input where additional arguments are passed through to <code>as.Date()</code> .
offset	[integer] or [date] Value you wish to start counting periods from relative to the Unix Epoch: <ul style="list-style-type: none"> • For integer values this is stored scaled by n (<code>offset <- as.integer(offset) %% n</code>). • For date values this is first converted to an integer offset (<code>offset <- floor(as.numeric(offset))</code>) and then scaled via n as above.

Value

A `<grates_period>` object.

Note

Internally `grates_period` objects are stored as the integer number, starting at 0L, of periods since the Unix Epoch (1970-01-01) and a specified offset. Here periods are taken to mean groupings of n consecutive days.

See Also

`as.Date()`

Examples

```
as_period("2019-05-03")
as_period("2019-05-03", n = 2, offset = 1)
as_period(as.POSIXct("2019-03-04 01:01:01", tz = "America/New_York"), n = 10)
as_period(as.Date("2020-03-02"), n = 2L, offset = as.Date("2020-03-01"))
```

as_year *Coerce an object to year-quarter*

Description

as_year() is a generic for coercing input in to <grates_year>.

Usage

```
as_year(x, ...)  
  
## Default S3 method:  
as_year(x, ...)  
  
## S3 method for class 'Date'  
as_year(x, ...)  
  
## S3 method for class 'POSIXt'  
as_year(x, ...)  
  
## S3 method for class 'character'  
as_year(x, ...)  
  
## S3 method for class 'factor'  
as_year(x, ...)
```

Arguments

x	R object. Character input is first parsed using as.Date(). POSIXct and POSIXlt are converted with the timezone respected.
...	Only used For character input where additional arguments are passed through to as.Date().

Value

A <grates_year> object.

See Also

as.Date()

Examples

```
as_year(Sys.Date())  
as_year(as.POSIXct("2019-03-04 01:01:01", tz = "America/New_York"), interval = 2)  
as_year("2019-05-03")
```

as_yearmonth	<i>Coerce an object to year-month</i>
--------------	---------------------------------------

Description

as_yearmonth() is a generic for coercing input in to <grates_yearmonth>. Character input is first parsed using as.Date(). POSIXct and POSIXlt are all converted, with the timezone respected.

Usage

```
as_yearmonth(x, ...)  
  
## Default S3 method:  
as_yearmonth(x, ...)  
  
## S3 method for class 'Date'  
as_yearmonth(x, ...)  
  
## S3 method for class 'POSIXt'  
as_yearmonth(x, ...)  
  
## S3 method for class 'character'  
as_yearmonth(x, ...)  
  
## S3 method for class 'factor'  
as_yearmonth(x, ...)
```

Arguments

x	R object.
...	Only used For character input where additional arguments are passed through to as.Date().

Value

A <grates_yearmonth> object.

Note

Internally <grates_yearmonth> objects are stored as the number of months (starting at 0) since the Unix Epoch (1970-01-01). Precision is only to the month level (i.e. the day of the month is always dropped).

References

The algorithm to convert between dates and months relative to the UNIX Epoch comes from the work of Davis Vaughan in the unreleased [datea](#) package.

See Also

as.Date()

Examples

```
as_yearmonth(Sys.Date())
as_yearmonth(as.POSIXct("2019-03-04 01:01:01", tz = "America/New_York"), interval = 2)
as_yearmonth("2019-05-03")
```

as_yearquarter	<i>Coerce an object to year-quarter</i>
----------------	---

Description

as_yearquarter() is a generic for coercing input in to <grates_yearquarter>. Character input is first parsed using as.Date(). POSIXct and POSIXlt are all converted, with the timezone respected.

Usage

```
as_yearquarter(x, ...)

## Default S3 method:
as_yearquarter(x, ...)

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

## S3 method for class 'POSIXt'
as_yearquarter(x, ...)

## S3 method for class 'character'
as_yearquarter(x, ...)

## S3 method for class 'factor'
as_yearquarter(x, ...)
```

Arguments

x	R object
...	Only used For character input where additional arguments are passed through to as.Date().

Value

A <grates_yearquarter> object.

Note

Internally `<grates_yearquarter>` objects are stored as the number of quarters (starting at 0) since the Unix Epoch (1970-01-01).

See Also

`as.Date()`

Examples

```
as_yearquarter(Sys.Date())
as_yearquarter(as.POSIXct("2019-03-04 01:01:01", tz = "America/New_York"), interval = 2)
as_yearquarter("2019-05-03")
```

as_yearweek	<i>Coerce to a yearweek object</i>
-------------	------------------------------------

Description

Generic for conversion to `<grates_yearweek>`.

Usage

```
as_yearweek(x, ...)

## Default S3 method:
as_yearweek(x, ...)

## S3 method for class 'Date'
as_yearweek(x, firstday = 1L, ...)

## S3 method for class 'POSIXt'
as_yearweek(x, firstday = 1L, ...)

## S3 method for class 'character'
as_yearweek(
  x,
  firstday = 1L,
  format,
  tryFormats = c("%Y-%m-%d", "%Y/%m/%d"),
  ...
)

## S3 method for class 'factor'
as_yearweek(
  x,
```

```

    firstday = 1L,
    format,
    tryFormats = c("%Y-%m-%d", "%Y/%m/%d"),
    ...
)

```

Arguments

x	R object.
...	Other values passed to as.Date().
firstday	[integer] The day the week starts on from 1 (Monday) to 7 (Sunday).
format	[character] Passed to as.Date() unless format = "yearweek" in which case input is assumed to be in the form "YYYY-Wxx". If not specified, it will try tryFormats one by one on the first non-NA element, and give an error if none works. Otherwise, the processing is via strptime() whose help page describes available conversion specifications.
tryFormats	[character] Format strings to try if format is not specified.

Details

- Date, POSIXct, and POSIXlt are converted with the timezone respected.
- Character objects are first coerced to date via as.Date() unless format = "yearweek" in which case input is assumed to be in the form "YYYY-Wxx" and parsed accordingly.

Value

A <grates_yearweek> object.

See Also

as.Date() and new_yearweek().

Examples

```

as_yearweek(Sys.Date())
as_yearweek(as.POSIXct("2019-03-04 01:01:01", tz = "America/New_York"))
as_yearweek("2019-05-03", firstday = 5L)
as_yearweek("2019-W12", format = "yearweek")

```

`boundaries`*Access the start (end) dates of a grates vector*

Description

Utility functions for accessing the start (end) dates for each element of a grates object and also checking whether a date is contained within that range

Usage

```
date_start(x)
```

```
date_end(x)
```

```
date %during% x
```

Arguments

`x` grouped date vector.

`date` A scalar <date> object.

Value

For `date_start` and `date_end` The requested start (end) dates for each element in the input. For `%during%` a logical vector indicating whether the date was present within the range of the tested object.

Examples

```
dates <- as.Date("2020-01-01") + 1:14
```

```
week <- as_isoweek(dates)
```

```
date_start(week)
```

```
date_end(week)
```

```
dates[1L] %during% week
```

```
period <- as_period(dates, n = 3)
```

```
date_start(period)
```

```
date_end(period)
```

```
dates[14L] %during% period
```

`epiweek`*Constructor for epiweek objects*

Description

`epiweek()` is a constructor for `<grates_epiweek>` objects.

Usage

```
epiweek(year = integer(), week = integer())
```

Arguments

<code>year</code>	<code>[integer]</code> Vector representing the year associated with week. double vectors will be converted via <code>as.integer(floor(x))</code> .
<code>week</code>	<code>[integer]</code> Vector representing the week associated with 'year'. double vectors will be converted via <code>as.integer(floor(x))</code> .

Details

Epiweeks are defined to start on a Sunday and `<grates_epiweek>` objects are stored as the number of weeks (starting at 0) from the first Sunday after the Unix Epoch (1970-01-01). That is, the number of seven day periods from 1970-01-04.

Internally they have the same representation as a `<grates_yearweek_sunday>` object so are akin to an alias but with a marginally more efficient implementation.

Value

A `<grates_epiweek>` object.

See Also

`as_epiweek()` and `new_epiweek()`.

Examples

```
epiweek(year = 2000L, week = 3L)
```

grouped_date_accessors
Accessors for grate objects

Description

Generics and methods for accessing information about grouped date objects.

Usage

```
get_firstday(x, ...)  
  
## Default S3 method:  
get_firstday(x, ...)  
  
## S3 method for class 'grates_yearweek_monday'  
get_firstday(x, ...)  
  
## S3 method for class 'grates_yearweek_tuesday'  
get_firstday(x, ...)  
  
## S3 method for class 'grates_yearweek_wednesday'  
get_firstday(x, ...)  
  
## S3 method for class 'grates_yearweek_thursday'  
get_firstday(x, ...)  
  
## S3 method for class 'grates_yearweek_friday'  
get_firstday(x, ...)  
  
## S3 method for class 'grates_yearweek_saturday'  
get_firstday(x, ...)  
  
## S3 method for class 'grates_yearweek_sunday'  
get_firstday(x, ...)  
  
get_week(x, ...)  
  
## Default S3 method:  
get_week(x, ...)  
  
## S3 method for class 'grates_yearweek'  
get_week(x, ...)  
  
## S3 method for class 'grates_epiweek'  
get_week(x, ...)
```

```
## S3 method for class 'grates_isoweek'  
get_week(x, ...)  
  
get_year(x, ...)  
  
## Default S3 method:  
get_year(x, ...)  
  
## S3 method for class 'grates_yearweek'  
get_year(x, ...)  
  
## S3 method for class 'grates_epiweek'  
get_year(x, ...)  
  
## S3 method for class 'grates_isoweek'  
get_year(x, ...)  
  
## S3 method for class 'grates_yearmonth'  
get_year(x, ...)  
  
## S3 method for class 'grates_yearquarter'  
get_year(x, ...)  
  
## S3 method for class 'grates_year'  
get_year(x, ...)  
  
get_n(x, ...)  
  
## Default S3 method:  
get_n(x, ...)  
  
## S3 method for class 'grates_month'  
get_n(x, ...)  
  
## S3 method for class 'grates_period'  
get_n(x, ...)  
  
## S3 method for class 'grates_int_period'  
get_n(x, ...)  
  
get_offset(x, ...)  
  
## Default S3 method:  
get_offset(x, ...)  
  
## S3 method for class 'grates_period'  
get_offset(x, ...)
```


Arguments

x	R object
...	Not currently used

Value

Requested value or an error if no method available.

Examples

```
dates <- as.Date("2020-01-01") + 1:14
dat <- as_isoweek(dates)
get_week(dat)
get_year(dat)
```

isoweek	<i>Constructor for isoweek objects</i>
---------	--

Description

`isoweek()` is a constructor for `<grates_isoweek>` objects.

Usage

```
isoweek(year = integer(), week = integer())
```

Arguments

year	[integer] Vector representing the year associated with week. double vectors will be converted via <code>as.integer(floor(x))</code> .
week	[integer] Vector representing the week associated with 'year'. double vectors will be converted via <code>as.integer(floor(x))</code> .

Details

isoweeks are defined to start on a Monday and `<grates_isoweek>` objects are stored as the number of weeks (starting at 0) from the first Monday prior to the Unix Epoch (1970-01-01). That is, the number of seven day periods from 1969-12-29.

Internally they have the same representation as a `<grates_yearweek_monday>` object so are akin to an alias but with a marginally more efficient implementation.

Value

A `<grates_isoweek>` object.

See Also

as_isoweek() and new_isoweek().

Examples

```
isoweek(year = 2000L, week = 3L)
```

new_epiweek

Minimal constructor for an epiweek object

Description

new_epiweek() is a constructor for <grates_epiweek> objects aimed at developers.

Usage

```
new_epiweek(x = integer())
```

```
is_epiweek(xx)
```

Arguments

x	[integer] Vector representing the number of weeks. double vectors will be converted via <code>as.integer(floor(x))</code> .
xx	R object.

Details

Epiweeks are defined to start on a Sunday and <grates_epiweek> objects are stored as the number of weeks (starting at 0) from the first Sunday after the Unix Epoch (1970-01-01). That is, the number of seven day periods from 1970-01-04.

Internally they have the same representation as a <grates_yearweek_sunday> object so are akin to an alias but with a marginally more efficient implementation.

Value

A <grates_epiweek> object.

See Also

new_yearweek() and new_isoweek().

Examples

```
new_epiweek(1:10)
```

`new_isoweek`*Minimal constructor for an isoweek object*

Description

`new_isoweek()` is a constructor for `<grates_isoweek>` objects aimed at developers.

Usage

```
new_isoweek(x = integer())
```

```
is_isoweek(xx)
```

Arguments

<code>x</code>	<code>[integer]</code> Vector representing the number of weeks. double vectors will be converted via <code>as.integer(floor(x))</code> .
<code>xx</code>	R object.

Details

isoweeks are defined to start on a Monday and `<grates_isoweek>` objects are stored as the number of weeks (starting at 0) from the first Monday prior to the Unix Epoch (1970-01-01). That is, the number of seven day periods from 1969-12-29.

Internally they have the same representation as a `<grates_yearweek_monday>` object so are akin to an alias but with a marginally more efficient implementation.

Value

A `<grates_isoweek>` object.

See Also

`new_yearweek()` and `new_epiweek()`.

Examples

```
new_isoweek(1:10)
```

`new_month`*Minimal Constructor for a month object*

Description

`new_month()` is a constructor for `<grates_month>` objects aimed at developers.

Usage

```
new_month(x = integer(), n)
```

```
is_month(xx)
```

Arguments

<code>x</code>	[integer] Vector representing the number of n-months since the Unix Epoch (1970-01-01). double vectors will be converted via <code>as.integer(floor(x))</code> .
<code>n</code>	[integer] Number of months that are being grouped. Must be greater than 1 (use <code>yearmonth()</code> for this case).
<code>xx</code>	R object.

Details

`grates_month` objects are stored as the integer number (starting at 0), of n-month groups since the Unix Epoch (1970-01-01). Here n-months is taken to mean a 'grouping of n consecutive months'.

Value

A `<grates_month>` object.

References

The algorithm to convert between dates and months relative to the UNIX Epoch comes from the work of Davis Vaughan in the unreleased `datea` package.

Examples

```
new_month(1:10, 2L)
```

new_period	<i>Minimal constructor for a period object</i>
------------	--

Description

new_period() is a constructor for <grates_period> objects aimed at developers.

Usage

```
new_period(x = integer(), n = 1L, offset = 0L)
```

```
is_period(xx)
```

Arguments

x	[integer] Vector representing the number of periods since the Unix Epoch (1970-01-01) and a specified offset. double vectors will be converted via <code>as.integer(floor(x))</code> .
n	[integer] Number of days that are being grouped by.
offset	[integer] Value you wish to start counting groups from relative to the Unix Epoch.
xx	R object.

Details

grates_period objects are stored as the integer number, starting at 0L, of periods since the Unix Epoch (1970-01-01) and a specified offset. Here periods are taken to mean groupings of n consecutive days.

For storage and calculation purposes, offset is scaled relative to n. I.e. `offset <- offset %/% n` and values of x stored relative to this scaled offset.

Value

A <grates_period> object.

Examples

```
new_period(1:10)
```

new_yearmonth	<i>Minimal constructor for a yearmonth object</i>
---------------	---

Description

`new_yearmonth()` is a constructor for `<grates_yearmonth>` objects aimed at developers.

Usage

```
new_yearmonth(x = integer())
```

```
is_yearmonth(xx)
```

Arguments

x	[integer] Vector representing the number of months. double vectors will be converted via <code>as.integer(floor(x))</code> .
xx	R object

Details

`<grates_yearmonth>` objects are stored as the number of months (starting at 0) since the Unix Epoch (1970-01-01). Precision is only to the month level (i.e. the day of the month is always dropped).

Value

A `<grates_yearmonth>` object.

References

The algorithm to convert between dates and months relative to the UNIX Epoch comes from the work of Davis Vaughan in the unreleased `datea` package

Examples

```
new_yearmonth(1:10)
```

new_yearquarter *Minimal constructor for a yearquarter object*

Description

new_yearquarter() is a constructor for <grates_yearquarter> objects aimed at developers.

Usage

```
new_yearquarter(x = integer())
```

```
is_yearquarter(xx)
```

Arguments

x	[integer] Vector representing the number of quarters. double vectors will be converted via <code>as.integer(floor(x))</code> .
xx	R object.

Details

<yearquarter> objects are stored as the number of quarters (starting at 0) since the Unix Epoch (1970-01-01).

Value

A <grates_yearquarter> object.

Examples

```
new_yearquarter(1:10)
```

new_yearweek *Minimal constructor for a yearweek object*

Description

new_yearweek() is a constructor for <grates_yearweek> objects aimed at developers.

Usage

```
new_yearweek(x = integer(), firstday = 1L)
```

```
is_yearweek(xx)
```

Arguments

<code>x</code>	[integer] Vector representing the number of weeks. double vectors will be converted via <code>as.integer(floor(x))</code> .
<code>firstday</code>	[integer] The day the week starts on from 1 (Monday) to 7 (Sunday).
<code>xx</code>	R object.

Details

`<grates_yearweek>` objects are stored as the number of weeks (starting at 0) from the date of the `firstday` nearest the Unix Epoch (1970-01-01). That is, the number of seven day periods from:

- 1969-12-29 for ``firstday`` equal to 1 (Monday)
- 1969-12-30 for ``firstday`` equal to 2 (Tuesday)
- 1969-12-31 for ``firstday`` equal to 3 (Wednesday)
- 1970-01-01 for ``firstday`` equal to 4 (Thursday)
- 1970-01-02 for ``firstday`` equal to 5 (Friday)
- 1970-01-03 for ``firstday`` equal to 6 (Saturday)
- 1970-01-04 for ``firstday`` equal to 7 (Sunday)

Value

A `<grates_yearweek>` object with subclass corresponding to the first day of the week they represent (e.g. `<grates_yearweek_monday>`).

See Also

`as_yearweek()`, `new_isoweek()` and `new_epiweek()`.

Examples

```
new_yearweek(1:10)
```

```
print.grates_int_period
```

Print an int_period object

Description

Print an `int_period` object

Usage

```
## S3 method for class 'grates_int_period'  
print(x, ...)  
  
## S3 method for class 'grates_int_period'  
format(x, ...)
```

Arguments

- x A <grates_int_period> object.
- ... Not currently used.

`print.grates_month` *Print a month object*

Description

Print a month object

Usage

```
## S3 method for class 'grates_month'  
print(x, format = "%Y-%b", sep = "to", ...)  
  
## S3 method for class 'grates_month'  
format(x, format = "%Y-%b", sep = "to", ...)
```

Arguments

- x A <grates_month> object.
- format [character]
 The format to use for the bounds of each value.
- sep [character]
 Where more than one month is grouped with others, sep is placed between the upper and lower bounds when printing.
- ... Not currently used.

print.grates_period *Print a period object*

Description

Print a period object

Usage

```
## S3 method for class 'grates_period'
print(x, format = "%Y-%m-%d", sep = "to", ...)

## S3 method for class 'grates_period'
format(x, format = "%Y-%m-%d", sep = "to", ...)
```

Arguments

x	A <grates_period> object.
format	[character] The format to use for the bounds of each value.
sep	[character] Where more than one day is grouped with others, sep is placed between the upper and lower bounds when printing.
...	Not currently used.

print.grates_year *Print a year-quarter object*

Description

Print a year-quarter object

Usage

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

## S3 method for class 'grates_year'
format(x, ...)
```

Arguments

x	A <grates_year> object.
...	Not currently used.

```
print.grates_yearmonth
```

Print a year-month object

Description

Print a year-month object

Usage

```
## S3 method for class 'grates_yearmonth'  
print(x, format = "%Y-%b", ...)
```

```
## S3 method for class 'grates_yearmonth'  
format(x, format = "%Y-%b", ...)
```

Arguments

x	A <grates_yearmonth> object.
format	The format to use for printing.
...	Not currently used.

```
print.grates_yearquarter
```

Print a year-quarter object

Description

Print a year-quarter object

Usage

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

```
## S3 method for class 'grates_yearquarter'  
format(x, ...)
```

Arguments

x	A <grates_yearquarter> object.
...	Not currently used.

scale_x_grates_epiweek
epiweek scale

Description

ggplot2 scale for an <grates_epiweek> vector.

Usage

```
scale_x_grates_epiweek(
  ...,
  breaks = ggplot2::waiver(),
  n.breaks = 6L,
  format = NULL
)
```

Arguments

...	Not currently used.
breaks	A <grates_epiweek> vector of the desired breaks.
n.breaks	[integer] Approximate number of breaks calculated using scales::breaks_pretty (default 6L). Will only have an effect if breaks = waiver().
format	Format to use if "Date" scales are required. If NULL (default) then labels are in the standard yearweek format (YYYY-Www). If not NULL then the value is used by format.Date() and can be any input acceptable by that function.

Value

A scale for use with ggplot2.

scale_x_grates_isoweek
isoweek scale

Description

ggplot2 scale for an <grates_isoweek> vector.

Usage

```
scale_x_grates_isoweek(
  ...,
  breaks = ggplot2::waiver(),
  n.breaks = 6L,
  format = NULL
)
```

Arguments

...	Not currently used.
breaks	A <grates_isoweek> vector of the desired breaks.
n.breaks	[integer] Approximate number of breaks calculated using scales::breaks_pretty (default 6L). Will only have an effect if breaks = waiver().
format	Format to use if "Date" scales are required. If NULL (default) then labels are in the standard yearweek format (YYYY-Www). If not NULL then the value is used by format.Date() and can be any input acceptable by that function.

Value

A scale for use with ggplot2.

scale_x_grates_month *month scale*

Description

ggplot2 scale for a month vector.

Usage

```
scale_x_grates_month(
  ...,
  breaks = ggplot2::waiver(),
  n.breaks = 6L,
  format = "%Y-%m-%d",
  bounds_format = "%Y-%b",
  sep = "to",
  n
)
```

Arguments

...	Not currently used.
breaks	A <grates_month> vector of the desired breaks.
n.breaks	[integer] Approximate number of breaks calculated using scales::breaks_pretty (default 6L). Will only have an effect if breaks = waiver().
format	Format to use if "Date" scales are required. If NULL then labels are centralised and of the form "lower category bound to upper category bound". If not NULL then the value is used by format.Date() and can be any input acceptable by that function (defaults to "%Y-%m-%d").
bounds_format	Format to use for grouped date labels. Only used if format is NULL.
sep	[character] Separator to use for grouped date labels.
n	[integer] Number of months used for the original grouping.

Value

A scale for use with ggplot2.

scale_x_grates_period *period scale*

Description

ggplot2 scale for an <grates_period> vector.

Usage

```
scale_x_grates_period(
  ...,
  breaks = ggplot2::waiver(),
  n.breaks = 6L,
  format = "%Y-%m-%d",
  n,
  offset
)
```

Arguments

...	Not currently used.
breaks	A <grates_period> vector of the desired breaks.
n.breaks	[integer] Approximate number of breaks calculated using scales::breaks_pretty (default 6L). Will only have an effect if breaks = waiver().
format	Format to use for dates. Value is used by format.Date() and can be any input acceptable by that function.
n	[integer] Number of days in each period.
offset	[integer] Number of days used in original grouping for the offset from the Unix Epoch.

Value

A scale for use with ggplot2.

scale_x_grates_year *year scale*

Description

ggplot2 scale for year vector.

Usage

```
scale_x_grates_year(
  ...,
  breaks = ggplot2::waiver(),
  n.breaks = 6L,
  format = NULL
)
```

Arguments

...	Not currently used.
breaks	A <grates_isoweek> vector of the desired breaks.
n.breaks	[integer] Approximate number of breaks calculated using scales::breaks_pretty (default 6L). Will only have an effect if breaks = waiver().
format	Format to use if "Date" scales are required. If not NULL then the value is used by format.Date() and can be any input acceptable by that function.

Value

A scale for use with `ggplot2`.

```
scale_x_grates_yearmonth
      yearmonth scale
```

Description

`ggplot2` scale for a yearmonth vector.

Usage

```
scale_x_grates_yearmonth(
  ...,
  breaks = ggplot2::waiver(),
  n.breaks = 6L,
  format = NULL
)
```

Arguments

<code>...</code>	Not currently used.
<code>breaks</code>	A <code><grates_yearmonth></code> vector of the desired breaks.
<code>n.breaks</code>	[integer] Approximate number of breaks calculated using <code>scales::breaks_pretty</code> (default 6L). Will only have an effect if <code>breaks = waiver()</code> .
<code>format</code>	Format to use if "Date" scales are required. If not NULL then the value is used by <code>format.Date()</code> and can be any input acceptable by that function.

Value

A scale for use with `ggplot2`.

scale_x_grates_yearquarter
yearquarter scale

Description

ggplot2 scale for a yearquarter vector.

Usage

```
scale_x_grates_yearquarter(
  ...,
  breaks = ggplot2::waiver(),
  n.breaks = 6L,
  format = NULL
)
```

Arguments

...	Not currently used.
breaks	A <grates_yearquarter> vector of the desired breaks.
n.breaks	[integer] Approximate number of breaks calculated using scales::breaks_pretty (default 6L). Will only have an effect if breaks = waiver().
format	Format to use if "Date" scales are required. If not NULL then the value is used by format.Date() and can be any input acceptable by that function.

Value

A scale for use with ggplot2.

scale_x_grates_yearweek
yearweek scale

Description

ggplot2 scale for an <grates_yearweek> vector.

Usage

```
scale_x_grates_yearweek(  
  ...,  
  breaks = ggplot2::waiver(),  
  n.breaks = 6L,  
  firstday,  
  format = NULL  
)  
  
scale_x_grates_yearweek_monday(  
  ...,  
  breaks = ggplot2::waiver(),  
  n.breaks = 6,  
  format = NULL  
)  
  
scale_x_grates_yearweek_isoweek(  
  ...,  
  breaks = ggplot2::waiver(),  
  n.breaks = 6,  
  format = NULL  
)  
  
scale_x_grates_yearweek_tuesday(  
  ...,  
  breaks = ggplot2::waiver(),  
  n.breaks = 6,  
  format = NULL  
)  
  
scale_x_grates_yearweek_wednesday(  
  ...,  
  breaks = ggplot2::waiver(),  
  n.breaks = 6,  
  format = NULL  
)  
  
scale_x_grates_yearweek_thursday(  
  ...,  
  breaks = ggplot2::waiver(),  
  n.breaks = 6,  
  format = NULL  
)  
  
scale_x_grates_yearweek_friday(  
  ...,  
  breaks = ggplot2::waiver(),  
  n.breaks = 6,
```

```

    format = NULL
  )

scale_x_grates_yearweek_saturday(
  ...,
  breaks = ggplot2::waiver(),
  n.breaks = 6,
  format = NULL
)

scale_x_grates_yearweek_sunday(
  ...,
  breaks = ggplot2::waiver(),
  n.breaks = 6,
  format = NULL
)

scale_x_grates_yearweek_epiweek(
  ...,
  breaks = ggplot2::waiver(),
  n.breaks = 6,
  format = NULL
)

```

Arguments

...	Not currently used.
breaks	A <grates_yearweek> vector of the desired breaks.
n.breaks	[integer] Approximate number of breaks calculated using <code>scales::breaks_pretty</code> (default 6L). Will only have an effect if <code>breaks = waiver()</code> .
firstday	[integer] Integer value of the first weekday: 1 (Monday) to 7 (Sunday).
format	Format to use if "Date" scales are required. If NULL (default) then labels are in the standard yearweek format (YYYY-Www). If not NULL then the value is used by <code>format.Date()</code> and can be any input acceptable by that function.

Value

A scale for use with `ggplot2`.

year	<i>Construct a year object</i>
------	--------------------------------

Description

year() is a constructor for <grates_year> objects.

Usage

```
year(x = integer())
```

```
is_year(object)
```

Arguments

x	[integer] Vector representing the years. double vectors will be converted via <code>as.integer(floor(x))</code> .
object	R object.

Value

A <grates_year> object.

Examples

```
year(2011:2020)
```

yearmonth	<i>Constructor for yearmonth objects</i>
-----------	--

Description

yearmonth() is a constructor for <grates_yearmonth> objects.

Usage

```
yearmonth(year = integer(), month = integer())
```

Arguments

year	[integer] Vector representing the year associated with month. double vectors will be converted via <code>as.integer(floor(x))</code> .
month	[integer] Vector representing the month associated with 'year'. double vectors will be converted via <code>as.integer(floor(x))</code> .

Details

<grates_yearmonth> objects are stored as the number of months (starting at 0) since the Unix Epoch (1970-01-01).

Value

A <grates_yearmonth> object.

See Also

as_yearmonth() and new_yearmonth().

Examples

```
yearmonth(year = 2000L, month = 3L)
```

yearquarter

Constructor for yearquarter objects

Description

yearquarter() is a constructor for <grates_yearquarter> objects.

Usage

```
yearquarter(year = integer(), quarter = integer())
```

Arguments

year	[integer] Vector representing the year associated with quarter. double vectors will be converted via <code>as.integer(floor(x))</code> .
quarter	[integer] Vector representing the quarter associated with 'year'. double vectors will be converted via <code>as.integer(floor(x))</code> .

Details

<grates_yearquarter> objects are stored as the number of quarters (starting at 0) since the Unix Epoch (1970-01-01).

Value

A <grates_yearquarter> object.

See Also

as_yearquarter() and new_yearquarter().

Examples

```
yearquarter(year = 2000L, quarter = 3L)
```

yearweek

Constructor for yearweek objects

Description

yearweek() is a constructor for <grates_yearweek> objects. These are weeks whose first day can be specified by the user.

Usage

```
yearweek(year = integer(), week = integer(), firstday = 1L)
```

Arguments

year	[integer]
	Vector representing the year associated with week. double vectors will be converted via <code>as.integer(floor(x))</code> .
week	[integer]
	Vector representing the week associated with 'year'. double vectors will be converted via <code>as.integer(floor(x))</code> .
firstday	[integer]
	The day the week starts on from 1 (Monday) to 7 (Sunday).

Details

For yearweek objects the first week of a "year" is considered to be the first yearweek containing 4 days of the given calendar year. This means that the calendar year will sometimes be different to that of the associated yearweek object.

Value

A <grates_yearweek> object with subclass corresponding to the first day of the week they represent (e.g. <grates_yearweek_monday>).

Note

Internally <grates_yearweek> objects are stored as the number of weeks (starting at 0) from the date of the firstday nearest the Unix Epoch (1970-01-01). That is, the number of seven day periods from:

- 1969-12-29 for `firstday` equal to 1 (Monday)
- 1969-12-30 for `firstday` equal to 2 (Tuesday)
- 1969-12-31 for `firstday` equal to 3 (Wednesday)
- 1970-01-01 for `firstday` equal to 4 (Thursday)
- 1970-01-02 for `firstday` equal to 5 (Friday)
- 1970-01-03 for `firstday` equal to 6 (Saturday)
- 1970-01-04 for `firstday` equal to 7 (Sunday)

See Also

as_yearweek() and new_yearweek().

Examples

```
yearweek(year = 2000L, week = 3L)
```

Index

`%during%` (boundaries), 13

`as_epiweek`, 2
`as_isoweek`, 4
`as_month`, 5
`as_period`, 6
`as_year`, 8
`as_yearmonth`, 9
`as_yearquarter`, 10
`as_yearweek`, 11

boundaries, 13

`date_end` (boundaries), 13
`date_start` (boundaries), 13

`epiweek`, 14

`format.grates_int_period`
 (`print.grates_int_period`), 24
`format.grates_month`
 (`print.grates_month`), 25
`format.grates_period`
 (`print.grates_period`), 26
`format.grates_year` (`print.grates_year`),
 26
`format.grates_yearmonth`
 (`print.grates_yearmonth`), 27
`format.grates_yearquarter`
 (`print.grates_yearquarter`), 27

`get_firstday` (`grouped_date_accessors`),
 15
`get_n` (`grouped_date_accessors`), 15
`get_offset` (`grouped_date_accessors`), 15
`get_week` (`grouped_date_accessors`), 15
`get_year` (`grouped_date_accessors`), 15
`grouped_date_accessors`, 15

`is_epiweek` (`new_epiweek`), 18
`is_isoweek` (`new_isoweek`), 19
`is_month` (`new_month`), 20
`is_period` (`new_period`), 21
`is_year` (`year`), 36
`is_yearmonth` (`new_yearmonth`), 22
`is_yearquarter` (`new_yearquarter`), 23
`is_yearweek` (`new_yearweek`), 23
`isoweek`, 17

`new_epiweek`, 18
`new_isoweek`, 19
`new_month`, 20
`new_period`, 21
`new_yearmonth`, 22
`new_yearquarter`, 23
`new_yearweek`, 23

`print.grates_int_period`, 24
`print.grates_month`, 25
`print.grates_period`, 26
`print.grates_year`, 26
`print.grates_yearmonth`, 27
`print.grates_yearquarter`, 27

`scale_x_grates_epiweek`, 28
`scale_x_grates_isoweek`, 28
`scale_x_grates_month`, 29
`scale_x_grates_period`, 30
`scale_x_grates_year`, 31
`scale_x_grates_yearmonth`, 32
`scale_x_grates_yearquarter`, 33
`scale_x_grates_yearweek`, 33
`scale_x_grates_yearweek_epiweek`
 (`scale_x_grates_yearweek`), 33
`scale_x_grates_yearweek_friday`
 (`scale_x_grates_yearweek`), 33
`scale_x_grates_yearweek_isoweek`
 (`scale_x_grates_yearweek`), 33
`scale_x_grates_yearweek_monday`
 (`scale_x_grates_yearweek`), 33

scale_x_grates_yearweek_saturday
 (scale_x_grates_yearweek), [33](#)
scale_x_grates_yearweek_sunday
 (scale_x_grates_yearweek), [33](#)
scale_x_grates_yearweek_thursday
 (scale_x_grates_yearweek), [33](#)
scale_x_grates_yearweek_tuesday
 (scale_x_grates_yearweek), [33](#)
scale_x_grates_yearweek_wednesday
 (scale_x_grates_yearweek), [33](#)

year, [36](#)
yearmonth, [36](#)
yearquarter, [37](#)
yearweek, [38](#)