Package 'grates'

May 8, 2026

Title Grouped Date Classes

Version 1.8.0

Description Provides a coherent interface and implementation for creating grouped date classes.

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

BugReports https://github.com/reconverse/grates/issues

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 3.6.0)

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

VignetteBuilder litedown

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

Config/testthat/edition 3

Config/testthat/start-first plots, refactoring

Imports utils, fastymd

NeedsCompilation no

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

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

Repository CRAN

Date/Publication 2026-03-25 15:30:02 UTC

Contents

`boundaries` *Access the bounding dates of a grates object*

Description

Utility functions for accessing the boundary dates for each element of a grates object.

Usage

```
date_start(x)

date_end(x)
```

Arguments

`x`	grouped date vector.

Value

The requested start and end dates for each element in the input.

Examples

```
dates <- as.Date("2020-01-01") + 1:9
week <- as_isoweek(dates)
date_start(week)
date_end(week)

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

---

epiweek_class *Epiweek class*

Description

Epiweeks are defined to start on a Sunday and span a 7 day period. Where they span calendar years, they are associated to the year which contains the majority of the week's days (i.e. the first epiweek a year is the one with at least four days in said year).

Internally, <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.

Usage

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

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"), ...)
```

```
new_epiweek(x = integer())

is_epiweek(xx)
```

Arguments

year	[integer] Vector representing the year associated with week. double vectors will be converted via as.integer(floor(x)).
week	[integer] Vector representing the week associated with 'year. double vectors will be converted via as.integer(floor(x)).
x, xx	R objects.
...	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

epiweek() is a constructor for <grates_epiweek> objects. It takes a vector of year and vector of week values as inputs. Length 1 inputs will be recycled to the length of the other input and double vectors will again be converted to integer via as.integer(floor(x)).

as_epiweek() is a generic for conversion to <grates_epiweek>.

- 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.

new_epiweek() is a minimal constructor for <grates_epiweek> objects aimed at developers. It takes, as input, the number of epiweeks since the sunday after the Unix Epoch that you wish to represent. double vectors will be converted to integer via as.integer(floor(x)).

Value

A <grates_epiweek> object.

See Also

The yearweek and isoweek classes.

Examples

```
# date coercion
as_epiweek(Sys.Date())

# POSIXt coercion
as_epiweek(as.POSIXct("2019-03-04 01:01:01", tz = "America/New_York"))

# character coercion assumes date input by default
as_epiweek("2019-05-03")

# character coercion can handle YYYY-Wxx format too
as_epiweek("2019-W12", format = "yearweek")

# construction
epiweek(year = 2000, week = 3)

# direct construction
stopifnot(
    identical(
        new_epiweek(0:1),
        as_epiweek("1970-01-04") + 0:1
    )
)
```

---

get_interval_duration *The number of days covered*

Description

Generic utility function for returning the number of days covered by an element of a grates object.

Usage

```
get_interval_duration(x, ...)
```

Arguments

x	A grouped date vector.
...	Not currently used.

Value

The number of days covered from the start to end dates for each element in the input.

Examples

```
# The following should be TRUE
identical(
    get_interval_duration(yearmonth(2020, 1:3)),
    c(31, 29, 31)
)
```

`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, ...)

## 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)
```

---

int_period_class *Integer-period class (Experimental)*

Description

<grates_int_period> objects represent groupings of n consecutive integers from 0.

Usage

```
## Default S3 method:
as_int_period(x, n = 1L, ...)

## S3 method for class 'integer'
as_int_period(x, n = 1L, ...)

## S3 method for class 'double'
as_int_period(x, n = 1L, ...)

new_int_period(x = integer(), n = 1L)

is_int_period(xx)

as_int_period(x, n, ...)
```

Arguments

x, xx	R objects. For as_int_period() this is the object to be coerced. For new_int_period() this represents the number of n integer periods from 0. double vectors will be converted via as.integer(floor(x)).
n	[integer] Number of integers that are being grouped. Must be greater than 0.
...	Not currently used.

Details

as_int_period() is a generic for coercing input into <grates_int_period> objects. For numeric input it coerces it's input x first via x <- as.integer(floor(x)) and then via integer division by n (i.e. x %/% n).

new_int_period() is a minimal constructor for <grates_period> objects aimed at developers. It takes, as input, the number of integer periods and the value of n.

Value

A <grates_int_period> object.

Examples

```
# coercion
as_int_period(1:10, n = 3)

# direct construction
stopifnot(
    identical(
        as_int_period(1:10, n = 3),
        new_int_period(c(0, 0, 1, 1, 1, 2, 2, 2, 3, 3), n = 3)
    )
)
```

---

isoweek_class *ISO Week class*

Description

<grates_isoweek> objects are used to represent ISO week dates as defined in ISO 8601. To expand further, it is easiest to quote from the related wikipedia entry:

"ISO weeks start with Monday and end on Sunday. Each week's year is the Gregorian year in which the Thursday falls. The first week of the year, hence, always contains 4 January. ISO week year numbering therefore usually deviates by 1 from the Gregorian for some days close to 1 January."

Internally, <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.

Usage

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

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"), ...)

new_isoweek(x = integer())

is_isoweek(xx)
```

Arguments

year	[integer] Vector representing the year associated with week. double vectors will be converted via as.integer(floor(x)).
week	[integer] Vector representing the week associated with 'year. double vectors will be converted via as.integer(floor(x)).
x, xx	R objects.
...	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

`isoweek()` is a constructor for `<grates_isoweek>` objects. It takes a vector of year and vector of week values as inputs. Length 1 inputs will be recycled to the length of the other input and `double` vectors will be converted to integer via `as.integer(floor(x))`.

`as_isoweek()` is a generic for conversion to `<grates_isoweek>`.

- 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.

`new_isoweek()` is a minimal constructor for `<grates_isoweek>` objects aimed at developers. It takes, as input, the number of isoweeks since the Monday prior to the Unix Epoch that you wish to represent. `double` vectors will be converted to integer via `as.integer(floor(x))`.

Value

A `<grates_isoweek>` object.

References

Wikipedia contributors. (2025, January 15). ISO week date. In Wikipedia, The Free Encyclopedia. Retrieved 09:19, March 6, 2025, from https://en.wikipedia.org/w/index.php?title=ISO_week_date&oldid=1269568343

See Also

The yearweek and epiweek classes.

Examples

```
# date coercion
as_isoweek(Sys.Date())

# POSIXt coercion
as_isoweek(as.POSIXct("2019-03-04 01:01:01", tz = "America/New_York"))

# character coercion assumes date input by default
as_isoweek("2019-05-03")

# character coercion can handle YYYY-Wxx format too
as_isoweek("2019-W12", format = "yearweek")

# construction
isoweek(year = 2000, week = 3)

# direct construction
stopifnot(
    identical(
        new_isoweek(0:1),
        as_isoweek("1969-12-29") + 0:1
    )
)
```

month_class *Month class*

Description

Month objects are groupings of 'n consecutive months' stored relative to the Unix Epoch. More precisely, grates_month objects are stored as the integer number (starting at 0), of n-month groups since the Unix Epoch (1970-01-01).

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, ...)

new_month(x = integer(), n)

is_month(xx)
```

Arguments

x, xx	R objects.
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().

Details

`as_month()` is a generic for conversion to `<grates_month>`.

- Character input is first parsed using `as.Date()`.
- POSIXt inputs are converted with the timezone respected.
- Precision is only to the month level (i.e. the day of the month is always dropped).

`new_month()` is a minimal constructor for `<grates_month>` objects aimed at developers. It takes, as input `x`, the number of n-months since the Unix Epoch (1970-01-01) and the related value of `n`. `double` vectors will be converted via `as.integer(floor(x))`.

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 datea package.

See Also

The yearmonth class.

Examples

```
# date coercion
as_month(Sys.Date(), n = 2)

# character coercion
as_month("2019-05-03", n = 4)

# POSIXt coercion
as_month(as.POSIXct("2019-03-04 01:01:01", tz = "America/New_York"), n = 2)

# direct construction
d <- seq.Date(from = as.Date("1970-03-01"), by = "2 month", length.out = 10)
stopifnot(
    identical(
        as_month(d, n = 2),
        new_month(1:10, 2)
    )
)
```

period_class *Period class*

Description

<grates_period> objects represent groupings of n consecutive days calculated relative to an offset. It is useful for when you wish to group an arbitrary number of dates together (e.g. 10 days).

Usage

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

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

## S3 method for class 'Date'
as_period(x, n = 1L, offset = min(x, na.rm = TRUE), ...)

## S3 method for class 'POSIXt'
as_period(x, n = 1L, offset = min(x, na.rm = TRUE), ...)

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

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

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

is_period(xx)
```

Arguments

x, xx	R objects. For as_period() this is the object to be coerced. For new_period() this represents the number of periods since the Unix Epoch (1970-01-01) and a specified offset.
n	[integer] Number of days that are being grouped.
...	Only used for character/factor input where additional arguments are passed through to as.Date().
offset	Value you wish to start counting periods from relative to the Unix Epoch: • For integer values this is stored scaled by n (offset <- as.integer(offset) %% n). • For date values this is first converted to an integer offset (offset <- floor(as.numeric(offset))) and then scaled via n as above.

- Character / factor inputs are first coerced to Dates and then handled accordingly.

Details

Internally grates_period objects are stored as the integer number, starting at 0, 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.

as_period() is a generic for coercing input into <grates_period> objects. It is the recommended way for constructing period objects as it allows the offset to be specified as a date (rather than an integer value relative to the Unix Epoch).

- Character input is first parsed using as.Date().
- POSIXct and POSIXlt are converted with their timezone respected.

new_period() is a minimal constructor for <grates_period> objects aimed at developers. It takes, as input, the number of periods since the Unix Epoch and the specified offset. double vectors will be converted via as.integer(floor(x)).

Value

A <grates_period> object.

Examples

```
# coercion from date
dat <- as.Date("2012-12-01")
as_period(dat + 0:3, n = 2, offset = dat)

# direct construction
new_period(1:10)
```

---

print.grates_month *Format and print a month object*

Description

Format and 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.

Value

For format(), a character vector representing the formatted input. print() is called for the side effect of printing to screen and thus returns the input <grates_month> object invisibly.

---

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.

Value

For format(), a character vector representing the formatted input. print() is called for the side effect of printing to screen and thus returns the input <grates_period> object invisibly.

print.grates_yearmonth *Format and print a yearmonth object*

Description

Format and print a yearmonth 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	[character] The format to use for printing.
...	Not currently used.

Value

For format(), a character vector representing the formatted input. print() is called for the side effect of printing to screen and thus returns the input <grates_yearmonth> object invisibly.

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" or year/week scales are required. If NULL (default) then labels are in the standard epiweek format (YYYY-Wxx). If "week" then the labels are of the form Www (e.g. W37). If "year" then labels are of the form YYYY (e.g. 2020). Otherwise the value is used by format.Date() and can be any input acceptable by that function. Note that when date scales are requested the grate labels are first coerced via as.Date().

Value

A scale for use with ggplot2.

Examples

```
# use simulated linelist data from the outbreaks package
linelist <- outbreaks::ebola_sim_clean$linelist
x <- as_epiweek(linelist$date_of_infection)
dat <- aggregate(list(cases = x), by = list(week = x), FUN = length)

# plot the output
(week_plot <-
    ggplot2::ggplot(dat, ggplot2::aes(week, cases)) +
    ggplot2::geom_col(width = 1, colour = "white") +
    ggplot2::theme_bw())

# We can have non-centred date labels on the x_axis by using the
# associated scale function and explicitly specifying a format for
# the date labels:
week_plot + scale_x_grates_epiweek(format = "%Y-%m-%d")
```

---

scale_x_grates_int_period

Integer-period scale (Experimental)

Description

ggplot2 scale for an integer-period vector.

Usage

```
scale_x_grates_int_period(
  ...,
  breaks = ggplot2::waiver(),
  n.breaks = 6L,
  centre = FALSE,
  n
)
```

Arguments

`...`	Not currently used.
`breaks`	A <grates_int_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()`.
`centre`	Only applicable to an `int_period` object with `n > 1`. If FALSE labels are place at the edge of the bounds. If TRUE then labels are centralised and of the form `[lower, upper]`
`n`	[integer] Number used for the original grouping.

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" or year/week scales are required. If NULL (default) then labels are in the standard isoweek format (YYYY-Wxx). If "week" then the labels are of the form Www (e.g. W37). If "year" then labels are of the form YYYY (e.g. 2020). Otherwise the value is used by format.Date() and can be any input acceptable by that function. Note that when date scales are requested the grate labels are first coerced via as.Date().

Value

A scale for use with ggplot2.

Examples

```
# use simulated linelist data from the outbreaks package
linelist <- outbreaks::ebola_sim_clean$linelist
x <- as_isoweek(linelist$date_of_infection)
dat <- aggregate(list(cases = x), by = list(week = x), FUN = length)

# plot the output
(week_plot <-
    ggplot2::ggplot(dat, ggplot2::aes(week, cases)) +
    ggplot2::geom_col(width = 1, colour = "white") +
    ggplot2::theme_bw())

# We can have non-centred date labels on the x_axis by using the
# associated scale function and explicitly specifying a format for
# the date labels:
week_plot + scale_x_grates_isoweek(format = "%Y-%m-%d")
```

---

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.

Examples

```
# use simulated linelist data from the outbreaks package
linelist <- outbreaks::ebola_sim_clean$linelist

# calculate the bimonthly number of cases
x <- as_month(linelist$date_of_infection, n = 2)
(dat <- aggregate(list(cases = x), by = list(group = x), FUN = length))

# by default lower date bounds are used for the x axis
(bimonth_plot <-
    ggplot2::ggplot(dat, ggplot2::aes(group, cases)) +
```

```
    ggplot2::geom_col(width = 1, colour = "white") +
    ggplot2::theme_bw() +
    ggplot2::theme(
        axis.text.x = ggplot2::element_text(
            angle = 45,
            hjust = 1
        )
    ) +
    ggplot2::xlab(""))

# To obtain centred labels you must explicitly set the format to NULL
# in the scale:
bimonth_plot + scale_x_grates_month(format = NULL, n = 2)
```

`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.

Examples

```
# use simulated linelist data from the outbreaks package
linelist <- outbreaks::ebola_sim_clean$linelist

# Calculate the total of infections across 14 day periods offset from
# the first date
x <- linelist$date_of_infection
x <- as_period(x, n = 14, offset = min(x, na.rm = TRUE))
dat <- aggregate(list(cases = x), by = list(period = x), FUN = length)
head(dat)

(period_plot <- ggplot2::ggplot(dat, ggplot2::aes(period, cases)) +
    ggplot2::geom_col(width = 1, colour = "white") +
    ggplot2::theme_bw() +
    ggplot2::theme(
        axis.text.x = ggplot2::element_text(
            angle = 45,
            hjust = 1
        )
    ) +
    ggplot2::xlab(""))

# To change defaults we must explicitly state the value of n and
# offset when calling the scale function
period_plot + scale_x_grates_period(
    n.breaks = 2,
    n = 14,
    offset = min(x, na.rm = TRUE)
)
```

---

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.

Examples

```
# use simulated linelist data from the outbreaks package
linelist <- outbreaks::ebola_sim_clean$linelist

# calculate yearly cases by date of infection
x <- as_year(linelist$date_of_infection)
(dat <- aggregate(list(cases = x), by = list(year = x), FUN = length))

# by default labels are centred
(year_plot <-
    ggplot2::ggplot(dat, ggplot2::aes(year, cases)) +
    ggplot2::geom_col(width = 1, colour = "white") +
    ggplot2::theme_bw() +
    ggplot2::xlab(""))

# To obtain centred labels you must explicitly set a date format
# in the scale:
year_plot + scale_x_grates_year(format = "%Y-%m-%d")
```

---

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

`...`	Not currently used.
`breaks`	A <grates_yearmonth> 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_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.

Examples

```
# use simulated linelist data from the outbreaks package
linelist <- outbreaks::ebola_sim_clean$linelist

# calculate quarterly cases by date of infection
x <- as_yearquarter(linelist$date_of_infection)
(dat <- aggregate(list(cases = x), by = list(quarter = x), FUN = length))

# by default labels are centred
(quarter_plot <-
    ggplot2::ggplot(dat, ggplot2::aes(quarter, cases)) +
    ggplot2::geom_col(width = 1, colour = "white") +
    ggplot2::theme_bw() +
    ggplot2::theme(
        axis.text.x = ggplot2::element_text(
            angle = 45,
            hjust = 1
        )
    ) +
    ggplot2::xlab(""))

# To obtain centred labels you must explicitly set a date format
# in the scale:
quarter_plot + scale_x_grates_yearquarter(format = "%Y-%m-%d")
```

---

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 `scales::breaks_pretty` (default 6L). Will only have an effect if `breaks = waiver()`.
`firstday`	[integer] Integer value of the first weekday: 1 (Monday) to 7 (Sunday).
`format`	Format to use if "Date" or year/week scales are required. If NULL (default) then labels are in the standard yearweek format (YYYY-Wxx). If "week" then the labels are of the form Www (e.g. W37). If "year" then labels are of the form YYYY (e.g. 2020). Otherwise the value is used by `format.Date()` and can be any input acceptable by that function. Note that when date scales are requested the grate labels are first coerced via `as.Date()`.

Value

A scale for use with ggplot2.

Examples

```
# use simulated linelist data from the outbreaks package
linelist <- outbreaks::ebola_sim_clean$linelist
x <- as_yearweek(linelist$date_of_infection, firstday = 4)
dat <- aggregate(list(cases = x), by = list(week = x), FUN = length)

# plot the output
(week_plot <-
    ggplot2::ggplot(dat, ggplot2::aes(week, cases)) +
    ggplot2::geom_col(width = 1, colour = "white") +
    ggplot2::theme_bw())

# We can have non-centred date labels on the x_axis by using the
# associated scale function and explicitly specifying a format for the
# date labels and a value for firstday:
week_plot + scale_x_grates_yearweek(format = "%Y-%m-%d", firstday = 4)
```

---

yearmonth_class *Yearmonth class*

Description

<grates_yearmonth> objects represent, unsurprisingly, years and associated months. Internally they 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).

Usage

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

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, ...)

new_yearmonth(x = integer())

is_yearmonth(xx)
```

Arguments

year	[integer] Vector representing the year associated with month. double vectors will be converted via as.integer(floor(x)).
month	[integer] Vector representing the month associated with year. double vectors will be converted via as.integer(floor(x)).
x, xx	R objects.
...	Only used for character input where additional arguments are passed through to as.Date().

Details

yearmonth() is a constructor for <grates_yearmonth> objects. It takes a vector of year and a vector of month values as inputs. Length 1 inputs will be recycled to the length of the other input and double vectors will be converted to integer via as.integer(floor(x)).

as_yearmonth() is a generic for coercing input into <grates_yearmonth>.

- Character input is first parsed using as.Date().
- POSIXct and POSIXlt are converted with their timezone respected.

new_yearmonth() is a minimal constructor for <grates_yearmonth> objects aimed at developers. It takes, as input, the number of months (starting at 0) since the Unix Epoch, that you wish to represent. double vectors will again be converted to integer via as.integer(floor(x)).

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

See Also

new_month() and as_month() and for grouping of consecutive months.

Examples

```
# date coercion
as_yearmonth(Sys.Date())

# POSIXt coercion
as_yearmonth(as.POSIXct("2019-03-04 01:01:01", tz = "America/New_York"))

# character coercion
as_yearmonth("2019-05-03")

# construction
yearmonth(year = 2000, month = 3)

# direct construction
d <- seq.Date(from = as.Date("1970-01-01"), by = "month", length.out = 10)
stopifnot(
    identical(
        as_yearmonth(d),
        new_yearmonth(0:9)
    )
)
```

`yearquarter_class` *Yearquarter class*

Description

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

Usage

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

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, ...)

new_yearquarter(x = integer())

is_yearquarter(xx)
```

Arguments

year	[integer] Vector representing the year associated with quarter. double vectors will be converted via as.integer(floor(x)).
quarter	[integer] Vector representing the quarter associated with year. double vectors will be converted via as.integer(floor(x)).
x, xx	R objects.
...	Only used for character input where additional arguments are passed through to as.Date().

Details

yearquarter() is a constructor for <grates_yearquarter> objects. It takes a vector of year and a vector of quarter values as inputs. Length 1 inputs will be recycled to the length of the other input and double vectors will be converted to integer via as.integer(floor(x)).

as_yearquarter() is a generic for coercing input into <grates_yearquarter>.

- Character input is first parsed using as.Date().
- POSIXct and POSIXlt are converted with their timezone respected.

new_yearquarter() is a minimal constructor for <grates_yearquarter> objects aimed at developers. It takes, as input, the number of quarters (starting at 0) since the Unix Epoch, that you wish to represent. double vectors will again be converted to integer via as.integer(floor(x)).

Value

A <grates_yearquarter> object.

Examples

```
# date coercion
as_yearquarter(Sys.Date())

# POSIXt coercion
as_yearquarter(as.POSIXct("2019-03-04 01:01:01", tz = "America/New_York"))

# character coercion
as_yearquarter("2019-05-03")
```

```
# construction
yearquarter(year = 2000, quarter = 3)

# direct construction
d <- seq.Date(from = as.Date("1970-01-01"), by = "quarter", length.out = 4)
stopifnot(
    identical(
        as_yearquarter(d),
        new_yearquarter(0:3)
    )
)
```

---

yearweek_class *Yearweek class*

Description

Yearweeks start on a user defined day of the week and span a 7 day period. 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.

Internally, <grates_yearweek> objects are stored as the number of weeks (starting at 0) from the date of the user-specified `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)
```

Usage

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

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"),
  ...
)

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

is_yearweek(xx)
```

Arguments

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

yearweek() is a constructor for <grates_yearweek> objects. These are weeks whose first day can be specified by the user. It takes a vector of year and vector of week values as inputs. Length 1 inputs will be recycled to the length of the other input and double vectors will again be converted to integer via as.integer(floor(x)).

as_yearweek() is a generic for conversion to <grates_yearweek>.

- 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.

new_yearweek() is a minimal constructor for <grates_yearweek> objects aimed at developers. It takes, as input, the number of weeks since the user-specified firstday nearest the Unix Epoch. double vectors will be converted to integer via as.integer(floor(x)).

Value

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

See Also

new_isoweek() and new_epiweek().

Examples

```
# date coercion
as_yearweek(Sys.Date())

# POSIXt coercion
as_yearweek(as.POSIXct("2019-03-04 01:01:01", tz = "America/New_York"))

# character coercion with Friday as the first day of the week
as_yearweek("2019-05-03", firstday = 5) # first day is Friday

# character coercion in yearweek format
as_yearweek("2019-W12", format = "yearweek")

# construction
yearweek(year = 2000, week = 3)

# direct construction
stopifnot(
    identical(
        new_yearweek(0:1, firstday = 1),
        as_yearweek("1969-12-29", firstday = 1) + 0:1
    )
)
```

`year_class` *Year class*

Description

Years are represented by a <grates_year> object.

Usage

```
year(x = integer())

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, ...)

is_year(object)
```

Arguments

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

Details

year() takes as input a vector representing, unsurprisingly, the years. double vectors are coerced via as.integer(floor(x)).

as_yearquarter() is a generic for coercing input into <grates_year>.

- Character input is first parsed using as.Date().
- POSIXct and POSIXlt are converted with their timezone respected.

Value

A <grates_year> object.

Examples

```
# date coercion
as_year(Sys.Date())

# POSIXt coercion
as_year(as.POSIXct("2019-03-04 01:01:01", tz = "America/New_York"))

# Character coercion
as_year("2019-05-03")

# direct construction
year(2011:2020)
```

`%during%` *Is a date covered by a grouped date*

Description

`%during%` determines whether a supplied date is within the period covered by each element of a grates object.

Usage

```
date %during% x
```

Arguments

`date`	A scalar <date> object.
`x`	grouped date vector.

Value

A logical vector indicating whether the date was present within the range of the tested object.

Examples

```
dates <- as.Date("2020-01-01") + 1:10
week <- as_isoweek(dates)
dates[1] %during% week

period <- as_period(dates, n = 3)
dates[10] %during% period
```

Index