

Package: betaCalendarsR (via r-universe)

October 11, 2026

Type Package

Title Deterministic Calendar Grids and Date Planning Utilities

Version 0.1.0

Description Tools for generating deterministic month, year, and date-range calendar structures from base R dates. Configurable week starts, fixed and compact month grids, adjacent-month cells, ISO week metadata, calendar boundaries, and bounded recurrence helpers support reporting, dashboards, scheduling interfaces, tests, and other applications that separate calendar structure from rendering.

License MIT + file LICENSE

Encoding UTF-8

URL <https://www.betacalendars.com/>,
<https://beta-calendar.r-universe.dev/betaCalendarsR>,
<https://github.com/Beta-Calendar/betaCalendarsR>

BugReports <https://github.com/Beta-Calendar/betaCalendarsR/issues>

Depends R (>= 4.1.0)

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

X-schema.org-keywords calendar, date-time, calendar-grid, date-range, planner, schedule, recurrence, year-month, reporting

Repository <https://beta-calendar.r-universe.dev>

Date/Publication 2026-10-06 10:42:35 UTC

RemoteUrl <https://github.com/Beta-Calendar/betaCalendarsR>

RemoteRef HEAD

RemoteSha 796fadf2c938c3b7ec4adefcfd9e78d8d80b119c

Contents

calendar_boundaries	2
calendar_sequence	3
calendar_summary	3
date_range_grid	4
last_weekday	4
month_grid	5
nth_weekday	6
recurrence_dates	6
validate_calendar_grid	7
year_grid	8
Index	9

calendar_boundaries	<i>Describe Boundaries in a Calendar Year</i>
---------------------	---

Description

Reports year and month endpoints, Gregorian leap-year facts, and ISO week-year crossover dates.

Usage

calendar_boundaries(year)

Arguments

year	Four-digit calendar year.
------	---------------------------

Value

A list of year dates, month endpoint tables, leap-year information, and ISO crossover dates.

Examples

calendar_boundaries(2027)\$iso_crossover_dates

calendar_sequence	Create a Sequence of Month Starts
-------------------	-----------------------------------

Description

Includes the month containing each endpoint of an inclusive date interval.

Usage

calendar_sequence(from, to)

Arguments

- from First date, inclusive.
- to Last date, inclusive.

Value

A data frame with one row per touched month and its first date.

Examples

```
calendar_sequence(as.Date("2026-12-20"), as.Date("2027-02-10"))
```

calendar_summary	Summarize Month-Grid Structure
------------------	--------------------------------

Description

Summarizes month lengths and the row counts used by calendar grids.

Usage

calendar_summary(year, month = NULL, week_start = "monday", fixed_rows = TRUE)

Arguments

- year Four-digit calendar year.
- month Optional month; omit to return all twelve.
- week_start First weekday in each row.
- fixed_rows Whether each grid uses six rows.

Value

A data frame with one row per requested month.

Examples

```
calendar_summary(2027, 2)
```

date_range_grid	<i>Build Calendar Metadata for a Finite Date Range</i>
-----------------	--

Description

Returns every date between two inclusive Date endpoints and adds calendar coordinates.

Usage

```
date_range_grid(from, to, week_start = "monday")
```

Arguments

- from First date, inclusive.
- to Last date, inclusive.
- week_start First weekday used for row and column coordinates.

Value

A data frame with one row for every consecutive date.

Examples

```
dates <- date_range_grid(as.Date("2026-12-30"), as.Date("2027-01-03"))
dates[c("date", "year", "month", "weekday", "iso_week_year")]
```

last_weekday	<i>Find a Month's Last Weekday</i>
--------------	------------------------------------

Description

Returns the last Monday-Friday date, or the final occurrence of a selected weekday.

Usage

```
last_weekday(year, month, weekday = NULL)
```

Arguments

- year Four-digit calendar year.
- month Month number from 1 through 12.
- weekday Optional full English weekday name.

Value

A Date.

Examples

```
last_weekday(2027, 1)
last_weekday(2027, 1, "monday")
```

month_grid	<i>Build a Deterministic Month Calendar Grid</i>
------------	--

Description

Creates complete calendar weeks for one month. Adjacent-month dates keep their positions so weekday columns remain stable.

Usage

```
month_grid(year, month, week_start = "monday", fixed_rows = TRUE, overflow = TRUE)
```

Arguments

year	Four-digit calendar year.
month	Month number from 1 through 12.
week_start	First weekday in each row; any full weekday name is accepted.
fixed_rows	If true, return six complete rows; otherwise return the minimum complete number of weeks.
overflow	If false, adjacent-month cells remain in the grid but have visible set to false.

Value

A data frame with one row per date cell, including date, calendar, ISO week, row/column, visibility, and boundary fields.

Examples

```
january <- month_grid(2027, 1, week_start = "monday")
nrow(january)
sum(january$in_month)
```

nth_weekday	<i>Find the Nth Weekday in a Month</i>
-------------	--

Description

Computes a selected occurrence without silently rolling an absent fifth weekday into the following month.

Usage

```
nth_weekday(year, month, weekday, n = 1L)
```

Arguments

year	Four-digit calendar year.
month	Month number from 1 through 12.
weekday	Full English weekday name.
n	Occurrence number from 1 through 5.

Value

A Date, or NA when that occurrence does not exist.

Examples

```
nth_weekday(2027, 1, "monday", 1)
```

recurrence_dates	<i>Generate Dates for a Bounded Recurrence Rule</i>
------------------	---

Description

Evaluates a weekly, monthly, nth-weekday, last-weekday, or annual date rule only within an explicit inclusive range.

Usage

```
recurrence_dates(
  from,
  to,
  frequency = c("weekly", "monthly", "nth_weekday", "last_weekday", "annual"),
  weekday = NULL,
  day = NULL,
  n = 1L,
  month_day = NULL,
  missing = c("skip", "clamp")
)
```

Arguments

from	First date, inclusive.
to	Last date, inclusive.
frequency	Recurrence kind.
weekday	Full English weekday name where used.
day	Day number for monthly rules.
n	Occurrence number for nth-weekday rules.
month_day	Annual month/day in MM-DD form.
missing	Skip or clamp a missing monthly or annual date.

Value

A date-ordered data frame with one row per recurrence and calendar metadata.

Examples

```
recurrence_dates(as.Date("2027-01-01"), as.Date("2027-01-31"),  
  frequency = "weekly", weekday = "monday")
```

`validate_calendar_grid`*Validate a Calendar-Grid Data Frame*

Description

Checks date, row, and column invariants for a calendar data frame.

Usage

```
validate_calendar_grid(x)
```

Arguments

x	A data frame returned by <code>month_grid</code> , <code>year_grid</code> , or <code>date_range_grid</code> .
---	---

Value

TRUE invisibly when valid; otherwise raises an informative error.

Examples

```
validate_calendar_grid(month_grid(2027, 1))
```

`year_grid`*Build Month Grids for a Calendar Year*

Description

Creates twelve ordered month grids with a shared week convention.

Usage

```
year_grid(  
  year,  
  week_start = "monday",  
  fixed_rows = TRUE,  
  overflow = TRUE,  
  as = c("data.frame", "list")  
)
```

Arguments

<code>year</code>	Four-digit calendar year.
<code>week_start</code>	First weekday in each row.
<code>fixed_rows</code>	Whether each month uses six rows.
<code>overflow</code>	Whether adjacent-month cells are visible.
<code>as</code>	Return a combined data frame or named list of month data frames.

Value

A January-to-December data frame or named list.

Examples

```
year <- year_grid(2027)  
table(year$month)  
months <- year_grid(2027, as = "list")
```

Index

`calendar_boundaries`, [2](#)
`calendar_sequence`, [3](#)
`calendar_summary`, [3](#)

`date_range_grid`, [4](#)

`last_weekday`, [4](#)

`month_grid`, [5](#)

`nth_weekday`, [6](#)

`recurrence_dates`, [6](#)

`validate_calendar_grid`, [7](#)

`year_grid`, [8](#)