Package 'shide'

November 4, 2026

Type Package

Title Date/Time Classes Based on Jalali Calendar

Version 0.4.0

Description Implements S3 classes for storing dates and date-times based on the Jalali calendar.
The main design goal of 'shide' is consistency with base R's 'Date' and 'POSIXct'.
It provide features such as: date-time parsing, formatting and arithmetic.

License MIT + file LICENSE

URL https://github.com/mmollayi/shide

BugReports https://github.com/mmollayi/shide/issues

Encoding UTF-8

LinkingTo cpp11 (>= 0.5.2), tzdb (>= 0.5.0)

Depends R (>= 4.3.0)

Imports cli, methods, rlang (>= 1.1.0), tzdb (>= 0.5.0), vctrs

Suggests covr, lubridate, pillar, testthat (>= 3.2.0)

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

NeedsCompilation yes

Author Mohsen Mollayi [aut, cre, cph]

Maintainer Mohsen Mollayi <mmollayi@gmail.com>

Repository CRAN

Date/Publication 2026-11-04 05:40:02 UTC

Contents

as_jdate 2
as_jdatetime 3
is_jdate 4
jdate 4
jdatetime 5

jdate_make . . . 7
jdate_now . . . 8
pretty.jdate . . . 9
seq.jdate . . . 10
seq.jdatetime . . . 11
shide-arithmetic . . . 13
shide-coercion . . . 14
shide-format . . . 15
shide-math . . . 17
sh_day . . . 18
sh_hour . . . 19
sh_labels . . . 21
sh_month . . . 22
sh_quarter . . . 23
sh_round . . . 23
sh_tzone . . . 24
sh_year . . . 25
sh_year_is_leap . . . 26
vec_cast.jdate . . . 27
vec_cast.jdatetime . . . 27

Index **29**

`as_jdate` *Cast an object to a* `jdate` *object*

Description

A generic function that converts other date/time classes to `jdate`.

Usage

```
as_jdate(x, ...)
```

Arguments

`x`	A vector of `jdatetime`, `POSIXct` or `Date`.
`...`	These dots are for future extensions and must be empty.

Details

Unlike R's `as.Date.POSIXct()` method, `as_jdate` does not expose a time zone argument and uses time zone attribute of input datetime object for conversion.

Value

A vector of `jdate` objects with the same length as x.

Examples

```
as_jdate(as.Date("2023-12-12"))
as_jdate(jdatetime("1402-09-21 13:14:00", tzone = "Asia/Tehran"))
as_jdate(as.POSIXct("2023-12-12 13:14:00", tz = "Asia/Tehran"))
```

---

as_jdatetime *Cast an object to a* jdatetime *object*

Description

A generic function that converts other date/time classes to jdatetime.

Usage

```
as_jdatetime(x, tzone, ...)
```

Arguments

x	a vector of jdate, POSIXct or Date.
tzone	A time zone name.
...	These dots are for future extensions and must be empty.

Details

If tzone is missing (default), time zone attribute of input object is used for conversion. If the input object does not have time zone attribute (e.g. jdate), and no value is supplied for tzone, local time zone is assumed for conversion.

Value

A vector of jdatetime objects with the same length as x.

Examples

```
## The time will be set to midnight when converting from `jdate` or `Date`
as_jdatetime(jdate_now())
as_jdatetime(Sys.Date())
## We can change time zone of a `jdatetime` to a new time zone
as_jdatetime(jdatetime_now(tzone = "Iran"), tzone = "Asia/Tokyo")
```

is_jdate *Check an object for its class*

Description

- `is_jdate()` checks whether an object is of class `jdate`.
- `is_jdatetime()` checks whether an object is of class `jdatetime`.

Usage

```
is_jdate(x)

is_jdatetime(x)
```

Arguments

x	An object to test.

Value

`TRUE` or `FALSE`.

Examples

```
is_jdate(jdate_now() + 1) # TRUE
is_jdatetime(jdatetime_now() + as.difftime(2, units = "hours")) # TRUE
```

jdate *Jalali calendar dates*

Description

`jdate` is an S3 class for representing the Jalali calendar dates. It can be constructed from character and numeric vectors.

Usage

```
jdate(x, ...)

## S3 method for class 'numeric'
jdate(x, ...)

## S3 method for class 'character'
jdate(x, format = NULL, ...)
```

Arguments

x	A vector of numeric or character objects.
...	Arguments passed on to further methods.
format	Format argument for character method.

Details

jdate is stored internaly as a double vector and doesn't have any attributes. Its value represents the count of days since the Unix epoch (a negative value if it represents a date prior to the epoch). This implementation coincides with the implementation of Date class.

Value

A vector of jdate objects.

Parsing limitations

Parsing and formatting are handled separately. While shide provides custom formatting with support for Persian month names and customizable labels, parsing currently relies on the underlying C++ date library and does not provide equivalent support for these features. Consequently, parsing and formatting are not fully symmetric.

Examples

```
jdate("1402-09-20")
jdate("1402/09/20", format = "%Y/%m/%d")
## Will replace invalid date format with NA
jdate("1402/09/20", format = "%Y-%m-%d")
## Invalid dates will be replaced with NA
jdate("1402-12-30")
## Jalali date corresponding to "1970-01-01"
jdate(0)
```

---

jdatetime *Date-time based on the Jalali calendar*

Description

jdatetime is an S3 class for representing date-times with the Jalali calendar dates. It can be constructed from character and numeric vectors.

Usage

```
jdatetime(x, tzone = "", ...)

## S3 method for class 'character'
jdatetime(x, tzone = "", format = NULL, ..., ambiguous = NULL)
```

Arguments

x	A vector of numeric or character objects.
tzone	A time zone name. Default value represents local time zone.
...	Arguments passed on to further methods.
format	Format argument for character method.
ambiguous	Resolve ambiguous times that occur during a repeated interval (when the clock is adjusted backwards during the transition from DST to standard time). Possible values are: • "earliest": Choose the earliest of the two moments. • "latest": Choose the latest of the two moments. • "NA": Produce NA. If NULL, defaults to "earliest"; as this seems to be base R's behavior.

Details

jdatetime is stored internally as a double vector and has a single attribute: the timezone (tzone). Its value represents the count of seconds since the Unix epoch (a negative value if it represents an instant prior to the epoch). This implementation coincides with that of POSIXct class, except that POSIXct may not have tzone attribute. But for jdatetime, tzone is not optional.

Value

A vector of jdatetime objects.

Parsing limitations

Parsing and formatting are handled separately. While shide provides custom formatting with support for Persian month names and customizable labels, parsing currently relies on the underlying C++ date library and does not provide equivalent support for these features. Consequently, parsing and formatting are not fully symmetric.

Examples

```
## default time zone and format
jdatetime("1402-09-20 18:57:09")
jdatetime("1402/09/20 18:57:09", tzone = "UTC", format = "%Y/%m/%d %H:%M:%S")
## Will replace invalid format with NA
jdatetime("1402/09/20 18:57:09", format = "%Y-%m-%d %H:%M:%S")
## nonexistent time will be replaced with NA
jdatetime("1401-01-02 00:30:00", tzone = "Asia/Tehran")
## ambiguous time will be replaced with NA
jdatetime("1401-06-30 23:30:00", tzone = "Asia/Tehran", ambiguous = "NA")
## ambiguous time will resolve by choosing the later time instant
jdatetime("1401-06-30 23:30:00", tzone = "Asia/Tehran", ambiguous = "latest")
## Jalali date-time in Iran time zone, corresponding to Unix epoch
jdatetime(0, "Iran")
```

jdate_make *Construct Jalali date-time objects from individual components*

Description

- `jdate_make()` creates a `jdate` object from individual components.
- `jdatetime_make()` creates a `jdatetime` object from individual components.

Usage

```
jdate_make(year, month = 1L, day = 1L, ...)

jdatetime_make(
  year,
  month = 1L,
  day = 1L,
  hour = 0L,
  minute = 0L,
  second = 0L,
  tzone = "",
  ...,
  ambiguous = NULL
)
```

Arguments

`year`	Numeric year.
`month`	Numeric month.
`day`	Numeric day.
`...`	These dots are for future extensions and must be empty.
`hour`	Numeric hour.
`minute`	Numeric minute.
`second`	Numeric second.
`tzone`	A time zone name. Default value represents local time zone.
`ambiguous`	Resolve ambiguous times that occur during a repeated interval (when the clock is adjusted backwards during the transition from DST to standard time). Possible values are: • `"earliest"`: Choose the earliest of the two moments. • `"latest"`: Choose the latest of the two moments. • `"NA"`: Produce `NA`. If `NULL`, defaults to `"earliest"`; as this seems to be base R's behavior.

Details

Numeric components are recycled to their common size using tidyverse recycling rules.

Value

- `jdate_make()` A vector of jdate object.
- `jdatetime_make()` A vector of jdatetime object.

Examples

```
## At least 'year' must be supplied
jdate_make(year = 1401)
## Components are recycled
jdatetime_make(year = 1399:1400, month = 12, day = c(30, 29), hour = 12, tzone = "UTC")
## resolve ambiguous time by choosing the later time instant
jdatetime_make(1401, 6, 30, 23, 0, 0, tzone = "Asia/Tehran", ambiguous = "latest")
```

---

jdate_now *Current Jalali date and datetime*

Description

System representation of the current time as jdate and jdatetime.

Usage

```
jdate_now(tzone = "")

jdatetime_now(tzone = "")
```

Arguments

`tzone`	Time zone to get the current time for.

Value

- `jdate_now()` A jdate object.
- `jdatetime_now()` A jdatetime object.

Examples

```
# Current Jalali date in the local time zone
jdate_now()

# Current Jalali date in a specified time zone
jdate_now("Asia/Tokyo")

# may be true or false
```

```
jdate_now("Asia/Tehran") == jdate_now("Asia/Tokyo")

# Current Jalali datetime in the local time zone
jdatetime_now()
```

`pretty.jdate` *Pretty breakpoints for Jalali date-times*

Description

Generates a sequence of about n+1 equally spaced nice, human-friendly time breaks which cover the given range.

Usage

```
## S3 method for class 'jdate'
pretty(x, n = 5, min.n = n%/%2, sep = " ", ..., labels = NULL)

## S3 method for class 'jdatetime'
pretty(x, n = 5, min.n = n%/%2, sep = " ", ..., labels = NULL)
```

Arguments

`x`	A vector of `jdate` or `jdatetime` objects.
`n`	Integer giving the desired number of intervals.
`min.n`	Non-negative integer giving the minimal number of intervals.
`sep`	A character string, serving as a separator for certain formats (e.g., between month and year).
`...`	These dots are for future extensions and must be empty.
`labels`	An object of class `shide_labels` created from `sh_labels()`

Value

A vector of Jalali date-times, with two additional attributes:

- "labels": A character vector of formatted labels corresponding to the breaks.
- "format": The format specification that was used.

Implementation notes

This method is implemented following the logic of `pretty.Date()` from the **grDevices** package in base R. It largely follows the behavior of `pretty.Date()` in determining breakpoints and constructing formatted labels. One intentional difference concerns the case `n = 0`. In this implementation:

- If `length(x) = 1`, the scalar input value is returned unchanged and the format specification is "%Y-%m-%d".
- If `length(x) > 1`, the function falls back to `pretty(x, n = 1)`.

See Also

grDevices::pretty.Date()

Examples

```
x <- seq(jdate("1404-01-02"), jdate("1404-03-03"), by = "week")
pretty(x, n = 5, min.n = 5)
# Use custom labels
labels <- sh_labels(month_abbr = paste0("Mon-", 1:12))
breaks <- pretty(x, labels = labels)
attr(breaks, "labels")

x <- jdatetime("1404-08-21 18:06:33", "Asia/Tehran")
pretty(x, n = 5)
```

---

seq.jdate *Generate regular sequences of Jalali dates*

Description

The method for seq for objects of class jdate.

Usage

```
## S3 method for class 'jdate'
seq(from, to, by, length.out = NULL, along.with = NULL, ...)
```

Arguments

from	starting date; optional.
to	end date; optional.
by	increment of the sequence; optional. See 'Details'.
length.out	integer, optional. Desired length of the sequence.
along.with	optional; take the length from the length of this argument.
...	arguments passed to or from other methods.

Details

by can be specified in several ways:

- A number, taken to be in days.
- A object of class difftime.
- A character string, containing one of "day", "week", "month", "quarter" or "year". This can optionally be preceded by a (positive or negative) integer and a space, or followed by "s".

Value

A vector of `jdate` objects.

Comparison with `seq.Date()`

The source code of `seq.jdate()` is a modified version of the code used in `base::seq.Date()`. But a few behaviors of the latter are changed:

- In base R, invalid dates resolve by overflowing according to the number of days that the date is invalid by. But `seq.jdate()` resolves invalid dates by rolling forward to the first day of the next month.
- If usage of `to` and `length.out` results in a fractional sequence between `from` and `to`, base R keeps the fraction in the underlying data of the output `Date` object. But since `jdate` is built upon whole numbers, the fractional part is dropped in the output.

These differences are illustrated in the examples.

See Also

`base::seq.Date()`

Examples

```
# by days
seq(jdate("1402-01-01"), jdate("1402-01-10"), 1)
# by 2 weeks
seq(jdate("1402-01-01"), jdate("1402-04-01"), "2 weeks")
# first days of years
seq(jdate("1390-01-01"), jdate("1399-01-01"), "years")
# by month
seq(jdate("1400-01-01"), by = "month", length.out = 12)
# quarters
seq(jdate("1400-01-01"), jdate("1403-01-01"), by = "quarter")

# fractional dates are allowed in `seq.Date()`, but not in `seq.jdate()`
unclass(seq(as.Date(0), as.Date(3), length.out = 3))
unclass(seq(jdate(0), jdate(2), length.out = 3))

# resloving strategy for invalid dates is different in 'seq.jdate()' compared to 'seq.Date()'
seq(as.Date("2021-01-31"), by = "months", length.out = 2)
seq(jdate("1402-06-31"), by = "6 months", length.out = 2)
```

`seq.jdatetime` *Generate regular sequences of Jalali date-times*

Description

The method for seq for objects of class `jdatetime`.

Usage

```
## S3 method for class 'jdatetime'
seq(from, to, by, length.out = NULL, along.with = NULL, ...)
```

Arguments

from	starting date; optional.
to	end date; optional.
by	increment of the sequence; optional. See 'Details'.
length.out	integer, optional. Desired length of the sequence.
along.with	take the length from the length of this argument.
...	arguments passed to or from other methods.

Details

by can be specified in several ways:

- A number, taken to be in seconds.
- An object of class difftime.
- A character string, containing one of "sec"/"second", "min"/"minute", "hour", "day", "DST-day","week", "month", "quarter" or "year". This can optionally be preceded by a (positive or negative) integer and a space, or followed by "s".

Value

A vector of jdatetime objects.

See Also

base::seq.POSIXt()

Examples

```
# first days of years
seq(jdatetime_make(1390, 1, 1), jdatetime_make(1399, 1, 1), "years")
# by month
seq(jdatetime_make(1400, 1, 1), by = "month", length.out = 12)
seq(jdatetime_make(1400, 1, 31), by = "month", length.out = 12)
# days vs DSTdays
seq(jdatetime_make(1400, 1, 1, 12, tzone = "Asia/Tehran"), by = "day", length.out = 2)
seq(jdatetime_make(1400, 1, 1, 12, tzone = "Asia/Tehran"), by = "DSTday", length.out = 2)
seq(jdatetime_make(1400, 1, 1, 12, tzone = "Asia/Tehran"), by = "1 week", length.out = 2)
seq(jdatetime_make(1400, 1, 1, 12, tzone = "Asia/Tehran"), by = "7 DSTdays", length.out = 2)
```

shide-arithmetic *Arithmetic operations for* jdate *and* jdatetime

Description

Arithmetic operations for jdate and jdatetime

Usage

```
## S3 method for class 'jdate'
vec_arith(op, x, y, ...)

## S3 method for class 'jdatetime'
vec_arith(op, x, y, ...)
```

Arguments

op	An arithmetic operator as a string.
x, y	A pair of vectors.
...	These dots are for future extensions and must be empty.

Details

Supported operations:

- Difference between two jdate objects results a difftime object with units = "days".
- Difference between two jdatetime objects results a difftime object with units = "seconds".
- A numeric vector can be added to or subtracted from a jdate or jdatetime.
- A difftime vector can be added to or subtracted from a jdate only if it has resolution bigger than "days".
- A difftime vector can be added to or subtracted from a jdatetime.
- A jdate object can be subtracted from a jdatetime and vice versa.

Value

The binary operator result of x and y. See details for more information on operator behaviors.

Examples

```
jdatetime_now() - jdate_now()
jdate_now() - as.difftime(1, units ="weeks" ) - as.difftime(1, units = "days")
```

shide-coercion *Coercion*

Description

Double dispatch methods to support `vctrs::vec_ptype2()`.

Usage

```
## S3 method for class 'jdate'
vec_ptype2(x, y, ..., x_arg = "", y_arg = "")

## S3 method for class 'jdatetime'
vec_ptype2(x, y, ..., x_arg = "", y_arg = "")
```

Arguments

x, y	Vector types.
...	These dots are for future extensions and must be empty.
x_arg, y_arg	Argument names for x and y. These are used in error messages to inform the user about the locations of incompatible types (see `stop_incompatible_type()`).

Details

Coercion rules for `jdate` and `jdatetime`:

- Combining a `jdate` and `jdatetime` yields a `jdatetime`.
- When combining two `jdatetime` objects, the timezone is taken from the first non-local timezone.

Value

An object prototype if x and y can be safely coerced to the same prototype; otherwise it returns an error. See details for more information on coercion hierarchy for `jdate` and `jdatetime`.

Examples

```
# jdate and jdatetime are compatible
c(jdate(), jdatetime(), jdatetime(tzone = "UTC"))
# jdate and Date are incompatible
try(c(jdate(), as.Date(NULL)))
```

shide-format *Format Jalali date-time objects*

Description

Format objects of class `jdate` and `jdatetime` into character vectors using a specified format string and optional localized labels.

Usage

```
## S3 method for class 'jdate'
format(x, format = NULL, ..., labels = NULL)

## S3 method for class 'jdatetime'
format(x, format = NULL, tz = NULL, ..., labels = NULL)
```

Arguments

`x`	A `jdate` or `jdatetime` vector.
`format`	A character string specifying the output format. If `NULL`, a default format is used. See "Format directives" for a complete list of supported directives.
`...`	These dots are for future extensions and must be empty.
`labels`	An object of class `shide_labels` created from `sh_labels()`
`tz`	Optional time zone for `jdatetime` objects. If provided, the date-time is formatted as if displayed in this time zone.

Value

A character vector representing the formatted dates or date-times.

Format directives

The `format` string may contain ordinary characters and formatting directives. Ordinary characters are copied unchanged to the output. Formatting directives begin with `%` and are replaced by the corresponding date or time component.

- %a: Abbreviated weekday name.
- %A: Full weekday name.
- %b: Abbreviated month name.
- %B: Full month name.
- %C: Year divided by 100 using floored division, zero-padded if the result is a single digit (00–99).
- %d: Day of the month as a zero-padded decimal number (01-31).
- %e: Day of the month as a space-padded decimal number (1-31).

- %F: Equivalent to %Y-%m-%d.
- %J: Equivalent to %Y/%m/%d.
- %H: Hour as a zero-padded decimal number using a 24-hour clock (00-23).
- %I: Hour as a zero-padded decimal number using a 12-hour clock (01-12).
- %j: Day of the year as a zero-padded decimal number (001-366).
- %m: Month as a zero-padded decimal number (01-12).
- %M: Minute as a zero-padded decimal number (00-59).
- %n: Newline character.
- %p: AM/PM indicator.
- %r: Equivalent to %I:%M:%S %p.
- %R: Equivalent to %H:%M.
- %S: Second as a zero-padded decimal number (00-59).
- %t: Tab character.
- %T: Equivalent to %H:%M:%S.
- %y: Last two decimal digits of the year, zero-padded if the result is a single digit (00-99).
- %Y: Year as a four-digits zero-padded decimal number (0000-9999).
- %z: Signed offset from UTC in the form `+HHMM` or `-HHMM`.
- %Z: Time zone abbreviation or name.
- %%: A % character.

See Also

`base::strftime()`

Examples

```
x <- jdate("1405-06-29")
format(x, "%Y/%m/%d")
format(x, "%J")
format(x, "%A, %d %B %Y")

# Use custom labels
labels <- sh_labels(
   weekday = c(
       "Shanbe", "Yekshanbe", "Doshanbe", "Seshanbe",
       "Chaharshanbe", "Panjshanbe", "Adine"
   )
)
format(x, "%A, %d %B %Y", labels = labels)

x <- jdatetime("1405-06-29 12:00:00", tzone = "Asia/Tehran")
# Format the same instant in different time zones.
format(x, "%F %T %Z (%z)")
format(x, "%F %T %Z (%z)", tz = "UTC")
```

shide-math *Mathematical operations for* jdate *and* jdatetime

Description

Math and Summary group of functions for jdate and jdatetime objects. Only methods for is.finite() and is.infinite() are provided and other functions from the groups, such as mean(), median() and summary() are not implemented.

Usage

```
## S3 method for class 'jdate'
vec_math(.fn, .x, ...)

## S3 method for class 'jdatetime'
vec_math(.fn, .x, ...)
```

Arguments

.fn	A mathematical function from the base package, as a string.
.x	A vector of jdate or jdatetime objects.
...	Additional arguments passed to .fn.

Details

vctrs implementation of Date and POSIXct does not include methods for is.finite() and is.infinite(). But these method are implemented in shide so that jdate and jdatetime vectors could be used as ggplot scales.

Value

For is.finite() and is.infinite(), a logical vector of the same length as x. Using all the other math and summary group generics will signal an error.

Examples

```
# Unlike a `Date` vector, `mean()` is not defined for a `jdate` vector
try(mean(c(jdate_now(), jdate_now() + 2)))
```

sh_day *Get/set the days components of Jalali date-time objects*

Description

- `sh_day()` and `sh_day<-()` retrieves and replaces the day of the month respectively.
- `mday()` and `mday<-()` are aliases for `day()` and `day<-()`.
- `sh_wday()` retrieves the day of the week.
- `sh_qday()` retrieves the day of the quarter.
- `sh_yday()` retrieves the day of the year.

Usage

```
sh_day(x)

sh_mday(x)

sh_wday(x)

sh_qday(x)

sh_yday(x)

## S3 method for class 'jdate'
sh_day(x)

## S3 method for class 'jdatetime'
sh_day(x)

## S3 method for class 'jdate'
sh_wday(x)

## S3 method for class 'jdatetime'
sh_wday(x)

## S3 method for class 'jdate'
sh_qday(x)

## S3 method for class 'jdatetime'
sh_qday(x)

## S3 method for class 'jdate'
sh_yday(x)

## S3 method for class 'jdatetime'
```

```
sh_yday(x)

sh_day(x) <- value

sh_mday(x) <- value

## S3 replacement method for class 'jdate'
sh_day(x) <- value

## S3 replacement method for class 'jdatetime'
sh_day(x) <- value
```

Arguments

x	A vector of jdate or jdatetime objects.
value	A numeric vector.

Details

For assignment, x and value are recycled to their common size using tidyverse recycling rules.

Value

The days component of x as an integer.

Examples

```
x <- jdate("1402-12-14")
sh_day(x)
sh_mday(x)
sh_qday(x)
sh_wday(x)
sh_yday(x)
sh_mday(x) <- 12:13
```

---

sh_hour *Get/set the time components of* jdatetime *objects*

Description

Get/set the time components of jdatetime objects

Usage

```
sh_hour(x)

sh_minute(x)

sh_second(x)

## S3 method for class 'jdatetime'
sh_hour(x)

## S3 method for class 'jdatetime'
sh_minute(x)

## S3 method for class 'jdatetime'
sh_second(x)

sh_hour(x) <- value

sh_minute(x) <- value

sh_second(x) <- value

## S3 replacement method for class 'jdatetime'
sh_hour(x) <- value

## S3 replacement method for class 'jdatetime'
sh_minute(x) <- value

## S3 replacement method for class 'jdatetime'
sh_second(x) <- value
```

Arguments

x	A vector of jdatetime objects.
value	A numeric vector.

Details

For assignment, x and value are recycled to their common size using tidyverse recycling rules.

Value

An integer vector representing the hour, minute or second component of x, depending on the function being called.

Examples

```
x <- jdatetime("1402-12-14 19:13:31")
sh_second(x)
```

```
sh_hour(x) <- 17:18
```

`sh_labels` *Create labels for date and time components*

Description

`sh_labels()` allows users to define custom textual representations for months, weekdays and AM/PM indicators, used by `shide` when parsing and formatting dates and times.

Usage

```
sh_labels(
  month = NULL,
  month_abbr = NULL,
  weekday = NULL,
  weekday_abbr = NULL,
  am_pm = NULL
)
```

Arguments

`month`, `month_abbr`	Optional character vectors of length 12 giving full and abbreviated month names, respectively, ordered from Farvardin to Esfand.
`weekday`, `weekday_abbr`	Optional character vectors of length 7 giving full and abbreviated weekday names, respectively, ordered from Saturday to Friday.
`am_pm`	An optional character vector of length 2 giving the AM and PM labels.

Value

An object of class `shide_labels` that stores labels for months, weekdays, and AM/PM.

Examples

```
# Create the default label set
labels <- sh_labels()

# Override multiple components
labels <- sh_labels(
   month = paste0("Month-", seq_len(12)),
   am_pm = c("morning", "evening")
)
```

sh_month *Get/set the month component of Jalali date-time objects*

Description

Get/set the month component of Jalali date-time objects

Usage

```
sh_month(x)

## S3 method for class 'jdate'
sh_month(x)

## S3 method for class 'jdatetime'
sh_month(x)

sh_month(x) <- value

## S3 replacement method for class 'jdate'
sh_month(x) <- value

## S3 replacement method for class 'jdatetime'
sh_month(x) <- value
```

Arguments

x	A vector of jdate or jdatetime objects.
value	A numeric vector.

Details

For assignment, x and value are recycled to their common size using tidyverse recycling rules.

Value

The month component of x as an integer.

Examples

```
x <- jdate("1402-12-14")
sh_month(x)
sh_month(x) <- 10:11
```

sh_quarter *Get the quarter of Jalali date-time objects*

Description

Get the quarter of Jalali date-time objects

Usage

```
sh_quarter(x)
```

Arguments

x	A vector of `jdate` or `jdatetime` objects.

Value

An integer vector of 1 to 4.

Examples

```
x <- jdate("1402-12-14")
sh_quarter(x)
```

sh_round *Round Jalali date-times to a specific unit of time*

Description

- `sh_floor()` takes a `jdate` or `jdatetime` object and rounds it down to the previous unit of time.
- `sh_ceiling()` takes a `jdate` or `jdatetime` object and rounds it up to the next unit of time.
- `sh_round()` takes a `jdate` or `jdatetime` object and and rounds it up or down, depending on what is closer. For dates or date-times which are exactly halfway between two consecutive units, the convention is to round up.

Usage

```
sh_round(x, unit = NULL, ...)

sh_floor(x, unit = NULL, ...)

sh_ceiling(x, unit = NULL, ...)
```

Arguments

x	A vector of jdate or jdatetime objects.
unit	A scalar character, containing a date or time unit or a multiple of a unit. Valid date units are "day", "week", "month", "quarter" and "year". Valid time units are "sec"/"second", "min"/"minute" and "hour". These can optionally be followed by "s". For jdate inputs, only date units may be supplied and for jdatetime inputs, both date and time units work. If multiple of a unit is used, unit coefficient must be a whole number greater than or equal to 1. If NULL, defaults to "day" for jdate inputs and"second" for jdatetime inputs.
...	These dots are for future extensions and must be empty.

Value

A vector of jdate or jdatetime objects with the same length as x.

See Also

lubridate::round_date()

Examples

```
x <- jdate("1402-12-15")
sh_floor(x, "year")
sh_floor(x, "2 months")
sh_ceiling(x, "year")
sh_round(x, "year")
sh_round(x, "week") == sh_floor(x, "week")
sh_round(x + 1, "week") == sh_ceiling(x, "week")

x <- jdatetime("1402-12-15 12:30:00", tzone = "Asia/Tehran")
sh_floor(x, "20 minutes")
sh_ceiling(x, "20 minutes")
sh_round(x, "1 hour") == sh_ceiling(x, "1 hour")
```

---

sh_tzone *Get the time zone component of* jdatetime *objects*

Description

Get the time zone component of jdatetime objects

Usage

```
sh_tzone(x)

## S3 method for class 'jdatetime'
sh_tzone(x)
```

Arguments

x	A vector of jdatetime objects.

Details

This function is only valid for jdatetime objects. jdate objects do not have time zone attribute.

Value

A character vector of length 1. An empty string represents the current local time zone.

Examples

```
x <- jdatetime("1402-12-14 19:13:31", tzone = "Iran")
sh_tzone(x)
```

---

sh_year *Get/set the year component of Jalali date-time objects*

Description

Get/set the year component of Jalali date-time objects

Usage

```
sh_year(x)

## S3 method for class 'jdate'
sh_year(x)

## S3 method for class 'jdatetime'
sh_year(x)

sh_year(x) <- value

## S3 replacement method for class 'jdate'
sh_year(x) <- value

## S3 replacement method for class 'jdatetime'
sh_year(x) <- value
```

Arguments

x	A vector of jdate or jdatetime objects.
value	A numeric vector.

Details

For assignment, `x` and `value` are recycled to their common size using tidyverse recycling rules.

Value

The year component of x as an integer.

Examples

```
x <- jdate("1402-12-14")
sh_year(x)
sh_year(x) <- 1400:1401
```

`sh_year_is_leap` *Determine if a Jalali year is a leap year*

Description

Check if an instant is in a leap year according to the Jalali calendar.

Usage

```
sh_year_is_leap(x)
```

Arguments

`x`	A `jdate` or `jdatetime` object or a numeric vector representing Jalali years.

Value

`TRUE` if in a leap year or `FALSE` otherwise.

Examples

```
sh_year_is_leap(jdatetime("1399-01-01 00:00:00"))
x <- seq(jdate("1400-01-01"), by = "years", length.out = 10)
names(x) <- sh_year(x)
sh_year_is_leap(x)
```

vec_cast.jdate *Cast an object to a* jdate *object*

Description

Cast an object to a jdate object

Usage

```
## S3 method for class 'jdate'
vec_cast(x, to, ...)
```

Arguments

x	Vectors to cast.
to	Type to cast to. If NULL, x will be returned as is.
...	For vec_cast_common(), vectors to cast. For vec_cast(), vec_cast_default(), and vec_restore(), these dots are only for future extensions and should be empty.

Value

A vector of jdate objects.

See Also

as_jdate is a convenience function that makes use of the casts that are defined for vec_cast.jdate() methods.

vec_cast.jdatetime *Cast an object to a* jdatetime *object*

Description

Cast an object to a jdatetime object

Usage

```
## S3 method for class 'jdatetime'
vec_cast(x, to, ...)
```

Arguments

x	Vectors to cast.
to	Type to cast to. If NULL, x will be returned as is.
...	For vec_cast_common(), vectors to cast. For vec_cast(), vec_cast_default(), and vec_restore(), these dots are only for future extensions and should be empty.

Value

A vector of jdatetime objects.

See Also

as_jdatetime is a convenience function that makes use of the casts that are defined for vec_cast.jdatetime() methods.

Index

as_jdate, 2, *27*
as_jdatetime, 3, *28*

base::seq.Date(), *11*
base::seq.POSIXt(), *12*
base::strftime(), *16*

difftime, *10*, *12*

format.jdate *(*shide-format*)*, 15
format.jdatetime *(*shide-format*)*, 15

grDevices::pretty.Date(), *10*

is_jdate, 4
is_jdatetime *(*is_jdate*)*, 4

jdate, 4
jdate_make, 7
jdate_now, 8
jdatetime, 5
jdatetime_make *(*jdate_make*)*, 7
jdatetime_now *(*jdate_now*)*, 8

lubridate::round_date(), *24*

pretty.jdate, 9
pretty.jdatetime *(*pretty.jdate*)*, 9

seq, *10, 11*
seq.jdate, 10
seq.jdatetime, 11
sh_ceiling *(*sh_round*)*, 23
sh_day, 18
sh_day<- *(*sh_day*)*, 18
sh_floor *(*sh_round*)*, 23
sh_hour, 19
sh_hour<- *(*sh_hour*)*, 19
sh_labels, 21
sh_labels(), *9*, *15*
sh_mday *(*sh_day*)*, 18
sh_mday<- *(*sh_day*)*, 18
sh_minute *(*sh_hour*)*, 19
sh_minute<- *(*sh_hour*)*, 19
sh_month, 22
sh_month<- *(*sh_month*)*, 22
sh_qday *(*sh_day*)*, 18
sh_quarter, 23
sh_round, 23
sh_second *(*sh_hour*)*, 19
sh_second<- *(*sh_hour*)*, 19
sh_tzone, 24
sh_wday *(*sh_day*)*, 18
sh_yday *(*sh_day*)*, 18
sh_year, 25
sh_year<- *(*sh_year*)*, 25
sh_year_is_leap, 26
shide-arithmetic, 13
shide-coercion, 14
shide-format, 15
shide-math, 17
stop_incompatible_type(), *14*

tidyverse recycling rules, *8*, *19, 20*, *22*, *26*

vctrs::vec_ptype2(), *14*
vec_arith.jdate *(*shide-arithmetic*)*, 13
vec_arith.jdatetime *(*shide-arithmetic*)*, 13
vec_cast.jdate, 27
vec_cast.jdatetime, 27
vec_math.jdate *(*shide-math*)*, 17
vec_math.jdatetime *(*shide-math*)*, 17
vec_ptype2.jdate *(*shide-coercion*)*, 14
vec_ptype2.jdatetime *(*shide-coercion*)*, 14