Package ‘era’

November 20, 2024

Type Package

Title Year-Based Time Scales

Version 0.5.0

Description Provides a consistent representation of year-based time scales as a numeric vector with an associated 'era'. There are built-in era definitions for many year numbering systems used in contemporary and historic calendars (e.g. Common Era, Islamic 'Hijri' years); year-based time scales used in archaeology, astronomy, geology, and other palaeosciences (e.g. Before Present, SI-prefixed 'annus'); and support for arbitrary user-defined eras. Years can converted from any one era to another using a generalised transformation function. Methods are also provided for robust casting and coercion between years and other numeric types, type-stable arithmetic with years, and pretty-printing in tables.

Language en-GB

License MIT + file LICENSE

URL https://era.joeroe.io, https://github.com/joeroe/era

BugReports https://github.com/joeroe/era/issues

Encoding UTF-8

RoxygenNote 7.3.2

Depends R (>= 2.10)

Imports vctrs (>= 0.3.0), methods, rlang, pillar

Suggests tibble, testthat, covr, knitr, rmarkdown, dplyr, spelling, purrr, tidyr, lubridate

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Joe Roe [aut, cre] (<https://orcid.org/0000-0002-1011-1244>)

Maintainer Joe Roe <joe@joeroe.io>

Repository CRAN

Date/Publication 2024-11-20 12:40:05 UTC

Contents

`era` *Create an era object*

Description

An `era` object defines the time scale associated with a vector of years (see `yr()`). `era()` returns an `era` object, either by looking up `label` in the standard eras defined in `eras()` or, if more than one argument is given, constructing a new definition with the specified parameters.

Usage

```
era(
  label = character(),
  epoch = NULL,
  name = label,
  unit = era_year("Gregorian"),
  scale = 1,
  direction = -1
)
```

Arguments

`label`	Character. If only one argument is given to `era()`, the abbreviated label of a standard era defined in `eras()`. Otherwise, the label to give to the era constructed using the following arguments.
`epoch`	Numeric. Epoch year from which years are counted in Gregorian astronomical years (i.e. there is a "year zero").
`name`	Character. Full name of the era. Defaults to the value of `label`.

`unit`	An `era_year()` object describing the name of the year unit and its average length in solar days. Defaults to a Gregorian year (365.2425 days).
`scale`	Integer. Number of years represented by one unit, e.g. `1000` for ka. Default: 1.
`direction`	Are years counted backwards (`-1`) (the default) or forwards (`1`) from `epoch`?

Value

An object of class `era`.

See Also

Other era definition functions: `eras()`

Other era helper functions: `era_parameters`, `era_year`, `era_year_parameters`, `is_era()`, `is_era_year()`, `is_yr()`, `this_year()`

Examples

```
era("cal BP")

era("T.A.", epoch = -9021, name = "Third Age", direction = 1)
```

`eras` *Standard era definitions*

Description

Definitions of common eras and time scales.

`eras()` lists all available era definitions. `eras(label)` looks up a specific era by its unique, abbreviated name (e.g. "cal BP").

Usage

```
eras(label = NA)
```

Arguments

`label`	(Optional) Abbreviated names(s) of eras to look up.

Details

Looking up eras by `label` uses partial matching.

Value

A table of era definitions. This can be passed to `era()` to construct an `era` object.

See Also

Other era definition functions: `era`

Examples

```
# List all available eras
eras()

# Look up a specific era by label
eras("cal BP")

# With partial matching
eras("cal")
```

`era_parameters` *Get parameters of an era*

Description

Extracts a specific parameter from an era object.

Usage

```
era_label(x)

era_epoch(x)

era_name(x)

era_unit(x)

era_scale(x)

era_direction(x)
```

Arguments

x	An `era` object.

Details

The available parameters are:

- **label** – unique, abbreviated label of the era, e.g. "cal BP"
- **epoch** – year of origin of the era, e.g. 1950 for Before Present
- **name** – full name of the era, e.g. "calendar years Before Present"
- **unit** – unit of years used, an `era_year()` object
- **scale** – multiple of years used, e.g. 1000 for ka/kiloannum
- **direction** – whether years are counted "backwards" or "forwards" from the epoch #'

Value

Value of the parameter.

See Also

Other era helper functions: `era`, `era_year`, `era_year_parameters`, `is_era()`, `is_era_year()`, `is_yr()`, `this_year()`

Examples

```
x <- era("cal BP")
era_name(x)
```

`era_year` *Year units*

Description

`era_year` objects describe the unit used for a year as its length in days. This value is used in an era definition (`era()`) to enable conversions between eras that use different units (with `yr_transform()`).

Usage

```
era_year(label = character(), days = 365.2425)
```

Arguments

`label`	Character. Name of the year unit.
`days`	Numeric. Average length of the year in solar days. Defaults to a Gregorian year (365.2425 days).

Value

S3 vector of class `era_year`.

See Also

Other era helper functions: `era`, `era_parameters`, `era_year_parameters`, `is_era()`, `is_era_year()`, `is_yr()`, `this_year()`

Examples

```
era_year("Julian", 365.25)
```

`era_year_parameters` *Get the parameters of an* `era_year` *object.*

Description

Extracts a specific parameter from a year unit object constructed by `era_year()`.

Usage

```
era_year_label(x)

era_year_days(x)
```

Arguments

`x`	An object of class `era_year`.

Value

Value of the parameter.

See Also

Other era helper functions: `era`, `era_parameters`, `era_year`, `is_era()`, `is_era_year()`, `is_yr()`, `this_year()`

Examples

```
julian <- era_year("Julian", 365.25)
era_year_label(julian)
era_year_days(julian)
```

`is_era` *Validation functions for* `era` *objects*

Description

Tests whether an object is an era definition (an `era` object). `is_era()` tests whether the object inherits from the S3 class `era_yr`. `is_valid_era()` performs additional checks to determine whether the object is well-formed (see details). `validate_era()` throws an informative error message for invalid `yrs`.

Usage

```
is_era(x)

validate_era(x)

is_valid_era(x)
```

Arguments

`x`	Object to test.

Details

Valid `era` objects:

- Must have all parameters set and not NA
- Must have a character `label` parameter
- Must have a numeric `epoch` parameter
- Must have a character `name` parameter
- Must have a character `unit` parameter that is one of the defined units
- Must have a positive, integer `scale` parameter
- Must have a direction parameter that is -1 (backwards) or 1 (forwards)

Value

`is_era()` and `is_valid_era()` return `TRUE` or `FALSE`. `validate_era()` returns `x` invisibly, and is used for its side-effect of throwing an informative error for invalid objects.

See Also

Other era helper functions: `era`, `era_parameters`, `era_year`, `era_year_parameters`, `is_era_year()`, `is_yr()`, `this_year()`

`is_era_year` *Validation functions for* `era_year` *objects*

Description

Tests whether an object is of class `era_year` (constructed by `era_year()`).

Usage

```
is_era_year(x)
```

Arguments

`x`	Object to test.

Value

`TRUE` or `FALSE`.

See Also

Other era helper functions: `era`, `era_parameters`, `era_year`, `era_year_parameters`, `is_era()`, `is_yr()`, `this_year()`

Examples

```
is_era_year(era_year("Julian", 365.25))
```

`is_yr` *Validation functions for* `yr` *objects*

Description

Tests whether an object is a vector of years with an era (a `yr` object). `is_yr()` tests whether the object inherits from the S3 class `era_yr`. `is_valid_yr()` performs additional checks to determine whether the object is well-formed (see details). `validate_yr()` throws an informative error message for invalid `yr`s.

Usage

```
is_yr(x)

validate_yr(x)

is_valid_yr(x)
```

Arguments

`x`	Object to test.

Details

Valid `yr` objects:

- Must contain numeric data (NAs are allowed)
- Must have the `era` attribute set and not NA
- Must not have more than one era
- Must have an `era` attribute that is a valid era object (see `validate_era()`)

Value

`is_yr()` and `is_valid_yr()` return `TRUE` or `FALSE`. `validate_yr()` returns `x` invisibly, and is used for its side-effect of throwing an informative error for invalid objects.

See Also

Other era helper functions: `era`, `era_parameters`, `era_year`, `era_year_parameters`, `is_era()`, `is_era_year()`, `this_year()`

Examples

```
x <- yr(5000:5050, era("cal BP"))
is_yr(x)
is_valid_yr(x)
validate_yr(x)
```

`this_year` *Current year*

Description

Returns the current year as a year vector, in the era system specified by `era`.

Usage

```
this_year(era = "CE")
```

Arguments

`era`	An `era` object or label understood by `era()`. Defaults to the Common Era (`era("CE")`).

Value

A `yr` vector with the current year.

See Also

Other era helper functions: `era`, `era_parameters`, `era_year`, `era_year_parameters`, `is_era()`, `is_era_year()`, `is_yr()`

Examples

```
# This year in the Common Era
this_year()
# This year in the Holocene Epoch
this_year("HE")
```

yr — *Create a vector of years with era*

Description

A yr object represents years with an associated calendar era or time scale.

Usage

```
yr(x = numeric(), era = character())
```

Arguments

x	A numeric vector of years.
era	The calendar era used by x. Either: • A string matching one of the standard era labels defined in eras() • An era object constructed with era()

Value

A yr (era_yr) object.

See Also

Other years with era functions: yr_era(), yr_transform()

Examples

```
# The R Age
yr(1993:2020, "CE")

# A bad movie
yr(10000, "BC")
```

yr_era — *Get or set the era of a vector of years*

Description

Functions for extracting or assigning the era of a vector of years. This function does not alter the underlying values of x. Use yr_transform() to *convert* the values of a yr vector to a new era.

Usage

```
yr_era(x)

yr_set_era(x, era)

yr_era(x) <- value
```

Arguments

`x`	A vector of years.
`value, era`	An `era` object (see `era()`) to be assigned to `x`.

Value

`yr_era(x)` returns the existing era associated with `x`.

`yr_set_era(x, era)` and `yr_era(x) <- era` return `x` with the new era assigned. If `x` is not already a `yr` vector, it will attempt to coerce it into one.

See Also

Other years with era functions: `yr()`, `yr_transform()`

Examples

```
x <- 5000:5050
yr_era(x) <- era("cal BP")
yr_era(x)
```

`yr_extremes` *Chronological minima and maxima*

Description

Returns the chronologically earliest and/or latest value in a vector of years, i.e. era-aware version `min()`, `max()`, and `range()`.

Usage

```
yr_earliest(x, na.rm = FALSE)

yr_latest(x, na.rm = FALSE)

yr_range(x, na.rm = FALSE)
```

Arguments

`x`	A yr vector with era
`na.rm`	a logical indicating whether missing values should be removed

Details

These are implemented as prefixed functions rather than S3 `min()`, `max()`, and `range()` methods for yrs to avoid surprises when numerical (i.e. not chronological) extremes are expected.

Value

For `yr_earliest()` and `yr_leatest()`, a `yr` vector of length 1 with the earliest or latest value.

For `yr_range()`, a `yr` vector of length 2 with the earliest and latest value (in that order).

If `x` contains `NA` values and `na.rm` = `FALSE` (the default), only `NA`s will be returned.

See Also

Other functions for chronological ordering and extremes: `yr_sort()`

Examples

```
# Forward-counting era:
x <- yr(c(200, 100, 300), "CE")
yr_earliest(x)
yr_latest(x)
yr_range(x)

# Backward-counting era:
y <- yr(c(200, 100, 300), "BCE")
yr_earliest(y)
yr_latest(y)
yr_range(x)
```

`yr_sort` *Chronological ordering of year vectors*

Description

Sorts a vector of years into earliest-to-latest or latest-to-earliest chronological order based on its era.

Usage

```
yr_sort(x, reverse = FALSE, ...)
```

Arguments

`x`	yr vector with era
`reverse`	Set `FALSE` (the default) for chronological order (earliest to latest) or `TRUE` for reverse chronological order (latest to earliest).
`...`	Other arguments passed to `sort()`; in particular use `na.last` to control NA handling.

Details

This is implemented as a prefixed function rather than an S3 `sort()` method for yrs to avoid surprises when numerical (i.e. not chronological) sorting is expected.

Value

Sorted yr vector

See Also

Other functions for chronological ordering and extremes: `yr_extremes`

Examples

```
# Forward-counting era:
x <- yr(c(200, 100, 300), "CE")
yr_sort(x)
yr_sort(x, reverse = TRUE)

# Backward-counting era:
y <- yr(c(200, 100, 300), "BCE")
yr_sort(y)
yr_sort(y, reverse = TRUE)
```

`yr_transform` *Convert years from one era to another*

Description

Transform a vector of years from one era to another.

Usage

```
yr_transform(x, era = yr_era(x), precision = NA)
```

Arguments

`x`	`yr` object. A vector of years with an era, see `yr()`.
`era`	`era` object describing the target era, see `era()`.
`precision`	Desired precision of the transformation, i.e. the transformed values are rounded to the nearest `precision`. If `NA` (the default), no rounding is performed and the exact transformed value is returned.

Details

Transformation between eras uses the `scale`, `epoch`, `direction` and `unit` parameters of the era definition. `NA` values for any of these parameters in the source or destination era will cause an error. This is most often encountered when either are measured in 'radiocarbon years', which cannot be related to a calendar time scale without calibration or un-calibration.

The transformation function is exact and treats years as a real number scale. This means that transformations between eras with different year units (e.g. Gregorian to Julian) and/or epochs not aligned to 1 January in the Gregorian calendar (e.g. Common Era to Islamic calendars) will likely return non-integer values. The `precision` argument provides a convenient way to round the result if you do not need this level of precision. It is also useful for working around the ambiguous definition of 'present' in various geological time-scales.

Value

A `yr` object in the era specified by `era`.

See Also

Other years with era functions: `yr()`, `yr_era()`

Examples

```
x <- yr(10010:10001, "cal BP")
yr_transform(x, era("BCE"))

yr_transform(x, era("ka"), precision = 1)
```

Index