Package: aion (via r-universe)

October 21, 2026

Title Archaeological Time Series

Version 1.7.1.9000

Maintainer Nicolas Frerebeau

<nicolas.frerebeau@u-bordeaux-montaigne.fr>

Description A toolkit for archaeological time series and time intervals. This package provides a system of classes and methods to represent and work with archaeological time series and time intervals. Dates are represented as ``rata die'' and can be converted to (virtually) any calendar defined by Reingold and Dershowitz (2018) <doi:10.1017/9781107415058>. This packages offers a simple API that can be used by other specialized packages.

License GPL (>= 3)

URL https://codeberg.org/tesselle/aion,
https://packages.tesselle.org/aion/,
https://tesselle.r-universe.dev/aion

BugReports https://codeberg.org/tesselle/aion/issues

Depends R (>= 3.5)

Imports arkhe (>= 1.10.0), graphics, grDevices, methods, stats, utils

Suggests folio (>= 1.5.0), fontquiver, igraph, knitr, markdown, relations, rsvg, svglite, tinysnapshot (>= 0.2.0), tinytest

VignetteBuilder knitr

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

X-schema.org-applicationCategory Archaeological Science

X-schema.org-isPartOf https://www.tesselle.org

X-schema.org-keywords time-series, time-intervals, chronology, stratigraphy, archaeology, archaeological-science, r-package

Collate 'AllClasses.R' 'AllGenerics.R' 'aion-internal.R' 'aion-package.R' 'axis.R' 'calendar-gregorian.R' 'calendar-julian.R' 'calendar.R' 'coerce.R' 'convert.R' 'data.R' 'format.R' 'graph.R' 'intervals.R' 'mutators.R' 'operators.R' 'overlap.R' 'plot.R' 'relations.R' 'series.R' 'show.R' 'span.R' 'subset.R' 'time.R' 'validate.R' 'year.R' 'zzz.R'

Config/roxygen2/version 8.0.0

Repository https://tesselle.r-universe.dev

Date/Publication 2026-06-23 10:41:09 UTC

RemoteUrl https://codeberg.org/tesselle/aion

RemoteRef HEAD

RemoteSha 2112caee7a38562c624dcd5914557aaae8870516

Contents

arithmetic . . . 3
as.data.frame . . . 4
as_date . . . 6
as_decimal . . . 7
as_fixed . . . 8
as_year . . . 9
calendar . . . 10
calendar_get . . . 12
convert . . . 14
dates . . . 15
fixed . . . 15
fixed_gregorian . . . 17
fixed_julian . . . 19
flip . . . 20
format . . . 21
get_calendar . . . 22
graph_create . . . 23
graph_prune . . . 25
gregorian . . . 27
image . . . 28
intervals . . . 29
is_calendar . . . 30
julian . . . 31
labels . . . 32
length . . . 32
names . . . 33
overlap . . . 34
plot . . . 35
pretty . . . 37
RataDie-class . . . 38

relations 39
series 43
span 45
start 46
subset 47
time 48
TimeIntervals-class 49
TimeScale-class 50
TimeSeries-class 51
window 51
year_axis 52

Index **54**

`arithmetic` *Arithmetic Operators*

Description

Operators performing arithmetic operations.

Usage

```
## S4 method for signature 'RataDie,RataDie'
Arith(e1, e2)

## S4 method for signature 'numeric,RataDie'
Arith(e1, e2)

## S4 method for signature 'RataDie,numeric'
Arith(e1, e2)
```

Arguments

`e1`, `e2`	A `RataDie` object or a `numeric` vector.

Details

Rata die will be converted to a plain `numeric` vector if a computation no longer makes sense in temporal terms.

Value

A `logical` vector.

Author(s)

N. Frerebeau

See Also

Other fixed date tools: as_date(), as_decimal(), as_fixed(), as_year(), fixed(), fixed_gregorian, fixed_julian, format(), pretty()

Examples

```
## Vectors of years
x <- fixed(c(-350, 31, 1072, 576, 1130), calendar = CE())
y <- fixed(c(1494, 1645, -869, 1440, 1851), calendar = CE())

## Move forward in time
x + y

## Move backward in time
x - y

## Not rata die anymore
x * y
```

---

as.data.frame *Coerce to a Data Frame*

Description

Coerce to a Data Frame

Usage

```
## S4 method for signature 'TimeSeries'
as.data.frame(x, ..., calendar = NULL)

## S4 method for signature 'TimeIntervals'
as.data.frame(x, ..., calendar = NULL)
```

Arguments

x	A TimeSeries or a TimeIntervals object.
...	Further parameters to be passed to data.frame().
calendar	A TimeScale object specifying the target calendar (see calendar()). If NULL (the default), *rata die* are returned.

Value

A data.frame.

Methods (by class)

- `as.data.frame(TimeSeries)`: Returns a long `data.frame` with the following columns:

  `time` The sampling times. If `calendar` is not `NULL`, it is expressed in decimal years; otherwise, it is expressed in *rata die*.

  `series` The name of the time series.

  `variable` The name of the variables.

  `value` The observed value.

- `as.data.frame(TimeIntervals)`: Returns a `data.frame` with the following columns:

  `label` The name of the intervals.

  `start` The start time of the intervals. If `calendar` is not `NULL`, it is expressed in decimal years; otherwise, it is expressed in *rata die*.

  `end` The end time of the intervals. If `calendar` is not `NULL`, it is expressed in decimal years; otherwise, it is expressed in *rata die*.

Author(s)

N. Frerebeau

Examples

```
## Create time-series of 20 observations

## Univariate
## Sampled every years starting from 1029 BCE
(X <- series(rnorm(30), time = 1029:1000, calendar = BCE()))

## Terminal and sampling times (returns rata die)
start(X)
end(X)
time(X)
span(X)

## Multivariate
## Sampled every century starting from 1000 CE
(Y <- series(matrix(rnorm(90), 30, 3), time = 1000:1029, calendar = CE()))

## Terminal and sampling times (returns Gregorian Common Era years)
start(Y, calendar = CE())
end(Y, calendar = CE())
time(Y, calendar = CE())
span(Y, calendar = CE())

## Coerce to data frame
df <- as.data.frame(Y, calendar = BP())
head(df)
```

as_date *Date Conversion from* Rata Die

Description

Date Conversion from *Rata Die*

Usage

```
as_date(object, calendar)

## S4 method for signature 'numeric,GregorianCalendar'
as_date(object, calendar)

## S4 method for signature 'numeric,JulianCalendar'
as_date(object, calendar)
```

Arguments

object	A RataDie object (see fixed()).
calendar	A TimeScale object specifying the target calendar (see calendar()).

Value

A numeric vector of (decimal) years.

Author(s)

N. Frerebeau

References

Reingold, E. M. and Dershowitz, N. (2018). *Calendrical Calculations: The Ultimate Edition*. Cambridge University Press. doi:10.1017/9781107415058.

See Also

Other fixed date tools: arithmetic, as_decimal(), as_fixed(), as_year(), fixed(), fixed_gregorian, fixed_julian, format(), pretty()

Examples

```
## R 1.0.0
(y <- fixed(year = 2000, month = 02, day = 29, calendar = CE()))
as_date(y, calendar = CE())
as_year(y, calendar = CE())

## Create a vector of years BP (Gregorian)
```

```
## (every two years starting from 2000 BP)
(years <- seq(from = 2000, by = -2, length.out = 10))
## Convert years to rata die
(rd <- fixed(years, calendar = BP()))
## Convert back to Gregorian years BP
as_year(rd, calendar = BP())

## More convenient
(rd <- fixed_from_BP(years))
fixed_to_BP(rd)
```

`as_decimal` *Converts a Date to a Decimal of its Year*

Description

Converts a Date to a Decimal of its Year

Usage

```
as_decimal(year, month, day, calendar)

## S4 method for signature 'numeric,numeric,numeric,GregorianCalendar'
as_decimal(year, month, day, calendar)

## S4 method for signature 'numeric,numeric,numeric,JulianCalendar'
as_decimal(year, month, day, calendar)
```

Arguments

`year`	A `numeric` vector of years. If `month` and `day` are missing, decimal years are expected.
`month`	A `numeric` vector of months.
`day`	A `numeric` vector of days.
`calendar`	A `TimeScale` object specifying the calendar of `year`, `month` and `day` (see `calendar()`).

Value

A `numeric` vector of (ecimal years.

Author(s)

N. Frerebeau

See Also

Other fixed date tools: `arithmetic`, `as_date()`, `as_fixed()`, `as_year()`, `fixed()`, `fixed_gregorian`, `fixed_julian`, `format()`, `pretty()`

Examples

```
## R 1.0.0
(y <- fixed(year = 2000, month = 02, day = 29, calendar = CE()))
as_date(y, calendar = CE())
as_year(y, calendar = CE())

## Create a vector of years BP (Gregorian)
## (every two years starting from 2000 BP)
(years <- seq(from = 2000, by = -2, length.out = 10))
## Convert years to rata die
(rd <- fixed(years, calendar = BP()))
## Convert back to Gregorian years BP
as_year(rd, calendar = BP())

## More convenient
(rd <- fixed_from_BP(years))
fixed_to_BP(rd)
```

---

as_fixed *Coerce to* Rata Die

Description

Coerce to *Rata Die*

Usage

```
as_fixed(from)

## S4 method for signature 'numeric'
as_fixed(from)
```

Arguments

from	A numeric vector of *rata die*.

Value

A RataDie object.

Author(s)

N. Frerebeau

References

Reingold, E. M. and Dershowitz, N. (2018). *Calendrical Calculations: The Ultimate Edition*. Cambridge University Press. doi:10.1017/9781107415058.

See Also

Other fixed date tools: arithmetic, as_date(), as_decimal(), as_year(), fixed(), fixed_gregorian, fixed_julian, format(), pretty()

Examples

```
## R 1.0.0
(y <- fixed(year = 2000, month = 02, day = 29, calendar = CE()))
as_date(y, calendar = CE())
as_year(y, calendar = CE())

## Create a vector of years BP (Gregorian)
## (every two years starting from 2000 BP)
(years <- seq(from = 2000, by = -2, length.out = 10))
## Convert years to rata die
(rd <- fixed(years, calendar = BP()))
## Convert back to Gregorian years BP
as_year(rd, calendar = BP())

## More convenient
(rd <- fixed_from_BP(years))
fixed_to_BP(rd)
```

---

as_year *Year Conversion from* Rata Die

Description

Year Conversion from *Rata Die*

Usage

```
as_year(object, calendar, ...)

## S4 method for signature 'numeric,GregorianCalendar'
as_year(object, calendar, decimal = TRUE, ...)

## S4 method for signature 'numeric,JulianCalendar'
as_year(object, calendar, decimal = FALSE, ...)
```

Arguments

object	A RataDie object (see fixed()).
calendar	A TimeScale object specifying the target calendar (see calendar()).
...	Currently not used.
decimal	A logical scalar: should decimal years be returned? If FALSE, the decimal part is dropped.

Value

A numeric vector of (decimal) years.

Author(s)

N. Frerebeau

References

Reingold, E. M. and Dershowitz, N. (2018). *Calendrical Calculations: The Ultimate Edition*. Cambridge University Press. doi:10.1017/9781107415058.

See Also

Other fixed date tools: arithmetic, as_date(), as_decimal(), as_fixed(), fixed(), fixed_gregorian, fixed_julian, format(), pretty()

Examples

```
## R 1.0.0
(y <- fixed(year = 2000, month = 02, day = 29, calendar = CE()))
as_date(y, calendar = CE())
as_year(y, calendar = CE())

## Create a vector of years BP (Gregorian)
## (every two years starting from 2000 BP)
(years <- seq(from = 2000, by = -2, length.out = 10))
## Convert years to rata die
(rd <- fixed(years, calendar = BP()))
## Convert back to Gregorian years BP
as_year(rd, calendar = BP())

## More convenient
(rd <- fixed_from_BP(years))
fixed_to_BP(rd)
```

---

calendar *Calendar*

Description

Calendar

Usage

```
calendar(object)

## S4 method for signature 'character'
calendar(object)
```

Arguments

`object`	A `character` string specifying the abbreviated label of the time scale (see details).

Details

The following time scales are available:

label	era	calendar
BP	Before Present	Gregorian
BC	Before Christ	Gregorian
BCE	Before Common Era	Gregorian
AD	Anno Domini	Gregorian
CE	Common Era	Gregorian
b2k	Years before 2000	Gregorian
julian		Julian

Value

A `TimeScale` object.

Note

Inspired by `era::era()` by Joe Roe.

Author(s)

N. Frerebeau

See Also

Other calendar tools: `calendar_get`, `convert()`, `get_calendar()`, `gregorian`, `is_calendar()`, `julian`

Examples

```
## Define time scales
calendar("BP")
calendar("AD")
calendar("julian")

## Shortcuts
BP()
AD()
J()
```

calendar_get *Calendar Parameters*

Description

Calendar Parameters

Usage

```
calendar_label(object)

calendar_name(object)

calendar_unit(object)

calendar_epoch(object)

calendar_fixed(object)

calendar_direction(object)

calendar_year(object)

## S4 method for signature 'TimeScale'
calendar_label(object)

## S4 method for signature 'TimeScale'
calendar_name(object)

## S4 method for signature 'TimeScale'
calendar_unit(object)

## S4 method for signature 'TimeScale'
calendar_epoch(object)

## S4 method for signature 'TimeScale'
calendar_fixed(object)

## S4 method for signature 'TimeScale'
calendar_direction(object)

## S4 method for signature 'NULL'
calendar_direction(object)

## S4 method for signature 'TimeScale'
calendar_year(object)
```

Arguments

object	A TimeScale object.

Value

- calendar_label() returns a character string giving the abbreviated label of the time scale.
- calendar_name() returns a character string giving the name of the time scale.
- calendar_unit() returns a character string giving the unit of the calendar.
- calendar_fixed() returns a length-one numeric vector giving the reference date of the calendar (in *rata die*).
- calendar_epoch() returns a length-one numeric vector giving the epoch year from which years are counted (starting date of the calendar, in years).
- calendar_direction() returns a length-one integer vector specifying if years are counted backwards (-1) or forwards (1) from epoch. Only the sign of calendar_direction() is relevant.
- calendar_year() returns a length-one numeric vector giving the average length of the year in solar days.

Author(s)

N. Frerebeau

See Also

Other calendar tools: calendar(), convert(), get_calendar(), gregorian, is_calendar(), julian

Examples

```
## Define time scales
calendar("BP")
calendar("AD")
calendar("julian")

## Shortcuts
BP()
AD()
J()
```

convert *Calendar Converter*

Description

Interconverts dates in a variety of calendars.

Usage

```
convert(from, to, ...)

## S4 method for signature 'character,character'
convert(from, to)

## S4 method for signature 'TimeScale,TimeScale'
convert(from, to)
```

Arguments

from	A TimeScale object describing the source calendar.
to	A TimeScale object describing the target calendar.
...	Currently not used.

Value

A function that when called with a single numeric argument (factional years) converts years from one calendar to another.

Author(s)

N. Frerebeau

See Also

Other calendar tools: calendar(), calendar_get, get_calendar(), gregorian, is_calendar(), julian

Examples

```
## Define time scales
BP <- calendar("BP")
AD <- calendar("AD")

## Make conversion functions
BP_to_AD <- convert(BP, AD)
AD_to_BP <- convert(AD, BP)

## Convert years
```

```
BP_to_AD(0)
AD_to_BP(1950)
```

`dates` *Sample Data from Reingold and Dershowitz (2018)*

Description

A dataset of 33 dates from the years -1000 to 2100 with their equivalents on differents calendars.

Usage

```
dates
```

Format

A `data.frame` with 33 rows and 14 variables:

`rata_die` Rata die.

`weekday` Week day.

`jd` Julian day.

`mjd` Modified Julian day.

`unix` Unix.

`gregorian_year`, `gregorian_month`, `gregorian_day` Gregorian date.

`julian_year`, `julian_month`, `julian_day` Julian date.

`egyptian_year`, `egyptian_month`, `egyptian_day` Egyptian date.

References

Reingold, E. M. and Dershowitz, N. (2018). *Calendrical Calculations: The Ultimate Edition*. Cambridge University Press. doi:10.1017/9781107415058.

`fixed` Rata Die *(Fixed Date)*

Description

Rata Die (Fixed Date)

Usage

```
fixed(year, month, day, calendar, ...)

## S4 method for signature 'numeric,missing,missing,GregorianCalendar'
fixed(year, calendar, scale = 1)

## S4 method for signature 'numeric,numeric,numeric,GregorianCalendar'
fixed(year, month, day, calendar)

## S4 method for signature 'numeric,missing,missing,JulianCalendar'
fixed(year, calendar, scale = 1)

## S4 method for signature 'numeric,numeric,numeric,JulianCalendar'
fixed(year, month, day, calendar)
```

Arguments

year	A numeric vector of years. If month and day are missing, decimal years are expected.
month	A numeric vector of months.
day	A numeric vector of days.
calendar	A TimeScale object specifying the calendar of year, month and day (see calendar()).
...	Currently not used.
scale	A length-one integer vector specifying the number of years represented by one unit. It should be a power of 10 (i.e. 1000 means ka).

Details

Rata die are represented as the number of days since 01-01-01 (Gregorian), with negative values for earlier dates.

Value

A RataDie object.

Author(s)

N. Frerebeau

References

Reingold, E. M. and Dershowitz, N. (2018). *Calendrical Calculations: The Ultimate Edition*. Cambridge University Press. doi:10.1017/9781107415058.

See Also

Other fixed date tools: arithmetic, as_date(), as_decimal(), as_fixed(), as_year(), fixed_gregorian, fixed_julian, format(), pretty()

Examples

```
## R 1.0.0
(y <- fixed(year = 2000, month = 02, day = 29, calendar = CE()))
as_date(y, calendar = CE())
as_year(y, calendar = CE())

## Create a vector of years BP (Gregorian)
## (every two years starting from 2000 BP)
(years <- seq(from = 2000, by = -2, length.out = 10))
## Convert years to rata die
(rd <- fixed(years, calendar = BP()))
## Convert back to Gregorian years BP
as_year(rd, calendar = BP())

## More convenient
(rd <- fixed_from_BP(years))
fixed_to_BP(rd)
```

`fixed_gregorian` Rata Die *Conversion to and from Gregorian Years*

Description

Convenient functions for conversion from and to *rata die* for a given Gregorian era.

Usage

```
fixed_from_BP(year, month, day)

fixed_to_BP(object)

fixed_from_BC(year, month, day)

fixed_to_BC(object)

fixed_from_BCE(year, month, day)

fixed_to_BCE(object)

fixed_from_AD(year, month, day)

fixed_to_AD(object)

fixed_from_CE(year, month, day)

fixed_to_CE(object)

fixed_from_b2k(year, month, day)
```

```
fixed_to_b2k(object)
```

Arguments

year	A numeric vector of years. If month and day are missing, decimal years are expected.
month	A numeric vector of months.
day	A numeric vector of days.
object	A RataDie object (see fixed()).

Details

The astronomical notation is used for Gregorian years (there *is* a year 0).

Value

- fixed_from_*() returns a RataDie object.
- fixed_to_*() returns a numeric vector of Gregorian years.

Author(s)

N. Frerebeau

References

Reingold, E. M. and Dershowitz, N. (2018). *Calendrical Calculations: The Ultimate Edition*. Cambridge University Press. doi:10.1017/9781107415058.

See Also

Other fixed date tools: arithmetic, as_date(), as_decimal(), as_fixed(), as_year(), fixed(), fixed_julian, format(), pretty()

Examples

```
## R 1.0.0
(y <- fixed(year = 2000, month = 02, day = 29, calendar = CE()))
as_date(y, calendar = CE())
as_year(y, calendar = CE())

## Create a vector of years BP (Gregorian)
## (every two years starting from 2000 BP)
(years <- seq(from = 2000, by = -2, length.out = 10))
## Convert years to rata die
(rd <- fixed(years, calendar = BP()))
## Convert back to Gregorian years BP
as_year(rd, calendar = BP())

## More convenient
```

```
(rd <- fixed_from_BP(years))
fixed_to_BP(rd)
```

`fixed_julian` Rata Die *Conversion to and from Julian Years*

Description

Convenient functions for conversion from and to *rata die*.

Usage

```
fixed_from_julian(year, month, day)

fixed_to_julian(object)
```

Arguments

`year`	A `numeric` vector of years. If `month` and `day` are missing, decimal years are expected.
`month`	A `numeric` vector of months.
`day`	A `numeric` vector of days.
`object`	A `RataDie` object (see `fixed()`).

Value

- `fixed_from_julian()` returns a `RataDie` object.
- `fixed_to_julian()` returns a `numeric` vector of Julian years.

Author(s)

N. Frerebeau

References

Reingold, E. M. and Dershowitz, N. (2018). *Calendrical Calculations: The Ultimate Edition*. Cambridge University Press. doi:10.1017/9781107415058.

See Also

Other fixed date tools: `arithmetic`, `as_date()`, `as_decimal()`, `as_fixed()`, `as_year()`, `fixed()`, `fixed_gregorian`, `format()`, `pretty()`

Examples

```
## R 1.0.0
(y <- fixed(year = 2000, month = 02, day = 29, calendar = CE()))
as_date(y, calendar = CE())
as_year(y, calendar = CE())

## Create a vector of years BP (Gregorian)
## (every two years starting from 2000 BP)
(years <- seq(from = 2000, by = -2, length.out = 10))
## Convert years to rata die
(rd <- fixed(years, calendar = BP()))
## Convert back to Gregorian years BP
as_year(rd, calendar = BP())

## More convenient
(rd <- fixed_from_BP(years))
fixed_to_BP(rd)
```

---

flip *Transposition*

Description

Transposes a time series object by permuting its dimensions and resizing it.

Usage

```
flip(x, ...)

## S4 method for signature 'TimeSeries'
flip(x)
```

Arguments

x	A $n \times m \times p$ TimeSeries object.
...	Currently not used.

Value

A permuted $(n \times p \times m)$ TimeSeries object.

Author(s)

N. Frerebeau

See Also

Other mutators: labels(), length, names

Examples

```
## Create 6 x 3 time-series of 50 observations
## Sampled every two years starting from 2000 BP
X <- series(
  object = array(rnorm(900), dim = c(50, 6, 3)),
  time = seq(2000, by = 2, length.out = 50),
  calendar = BP()
)
plot(X, calendar = BP())

## Permute dimensions 2 and 3
## 3 x 6 time-series of 50 observations
Y <- flip(X)
plot(Y, calendar = BP())
```

`format` *Date Conversion to Character*

Description

Date Conversion to Character

Usage

```
## S4 method for signature 'TimeIntervals'
format(x, calendar = get_calendar(), ...)

## S4 method for signature 'TimeScale'
format(x, ...)

## S4 method for signature 'RataDie'
format(
  x,
  prefix = c("a", "ka", "Ma", "Ga"),
  label = TRUE,
  calendar = get_calendar(),
  ...
)
```

Arguments

`x`	A `RataDie` object.
`calendar`	A `TimeScale` object specifying the target calendar (see `calendar()`).
`...`	Currently not used.
`prefix`	A `character` string specifying the prefix. It should be one of `"a"`, `"ka"`, `"Ma"` or `"Ga"`. If `TRUE`, a good guess for an appropriate format is made.
`label`	A `logical` scalar: should the label of the calendar be displayed?

Value

A character vector representing the date.

Author(s)

N. Frerebeau

See Also

Other fixed date tools: arithmetic, as_date(), as_decimal(), as_fixed(), as_year(), fixed(), fixed_gregorian, fixed_julian, pretty()

Examples

```
## R 1.0.0
(y <- fixed(year = 2000, month = 02, day = 29, calendar = CE()))
as_date(y, calendar = CE())
as_year(y, calendar = CE())

## Create a vector of years BP (Gregorian)
## (every two years starting from 2000 BP)
(years <- seq(from = 2000, by = -2, length.out = 10))
## Convert years to rata die
(rd <- fixed(years, calendar = BP()))
## Convert back to Gregorian years BP
as_year(rd, calendar = BP())

## More convenient
(rd <- fixed_from_BP(years))
fixed_to_BP(rd)
```

---

get_calendar *Get or Set the Default Calendar*

Description

Get or Set the Default Calendar

Usage

```
get_calendar(which = c("default", "current"))

set_calendar(x, which = c("default", "current"))
```

Arguments

which	A character string specifying the calendar to be set. It must be one of "default" or "current". Note that "current" is automatically set by plot() or image() and should not be changed manually.
x	A character string specifying the abbreviated label of the time scale (see calendar()) or an object from which to extract the time scale.

Value

A TimeScale object.

Author(s)

N. Frerebeau

See Also

Other calendar tools: calendar(), calendar_get, convert(), gregorian, is_calendar(), julian

Examples

```
## Define time scales
calendar("BP")
calendar("AD")
calendar("julian")

## Shortcuts
BP()
AD()
J()
```

---

graph_create *Create a Graph*

Description

Creates an interval or a stratigraphic graph.

Usage

```
graph_create(object, ...)

## S4 method for signature 'data.frame'
graph_create(
  object,
  type = c("interval", "stratigraphy"),
  direction = c("above", "below"),
  verbose = getOption("aion.verbose"),
```

```
  ...
)

## S4 method for signature 'matrix'
graph_create(
  object,
  type = c("interval", "stratigraphy"),
  direction = c("above", "below"),
  verbose = getOption("aion.verbose"),
  ...
)

## S4 method for signature 'TimeIntervals'
graph_create(
  object,
  type = c("interval", "stratigraphy"),
  verbose = getOption("aion.verbose"),
  ...
)
```

Arguments

object	A TimeIntervals object or a two-columns character matrix of edges (i.e. where each row specifies one relation element).
...	Currently not used.
type	A character string specifying the type of the graph to be computed. It must be one of "interval" (the default) or "stratigraphy" (see details). Any unambiguous substring can be given.
direction	A character string specifying the direction of the relations in x. It must be one of "above" (the default) or "below" (see details). Any unambiguous substring can be given. Only relevant if type is "stratigraphy".
verbose	A logical scalar: should R report extra information on progress?

Details

`interval` An interval graph is the graph showing intersecting intervals on a line. As time is linear and not circular, an interval graph contains no cycles with more than three edges and no shortcuts (it must be a chordal graph).

`stratigraphy` A stratigraphic graph represents the directed relationships between temporal units (archaeological deposits), from the most recent to the oldest (Harris 1997). It can be formally defined as a directed acyclic graph (DAG), in which each vertex represents a layer and the edges represent stratigraphic relations.

Value

An **igraph** graph object.

Note

Experimental.

The **igraph** and **relations** packages need to be installed on your machine.

Author(s)

N. Frerebeau

References

Harris, Edward C., 1997. *Principles of Archaeological Stratigraphy*. Seconde edition. Academic Press.

See Also

Other graph tools: `graph_prune()`

Examples

```
if (requireNamespace("igraph", quietly = TRUE) &&
    requireNamespace("relations", quietly = TRUE)) {
  ## Seven intervals
  int <- intervals(
    start = c(1, 2, 3, 6, 9, 13, 17),
    end = c(7, 4, 15, 14, 11, 18, 19),
    calendar = CE(),
    names = c("A", "B", "C", "D", "E", "F", "G")
  )

  ## Interval graph
  g <- graph_create(int, type = "interval")
  plot(g)

  ## Stratigraphic graph
  g <- graph_create(int, type = "strati")
  g <- graph_prune(g) # Remove redundant relations
  plot(g, layout = igraph::layout_with_sugiyama)
}
```

---

graph_prune *Prune a Graph*

Description

Removes redundant relations from a graph.

Usage

```
graph_prune(object, ...)

## S4 method for signature 'igraph'
graph_prune(
  object,
  reduce = TRUE,
  remove_multiple = TRUE,
  remove_loops = TRUE,
  ...
)
```

Arguments

object	An **igraph** object (typically returned by graph_create()).
...	Currently not used.
reduce	A logical scalar: should transitive reduction be performed? Only used if object is a directed acyclic graph.
remove_multiple	A logical scalar: should multiple edges be removed?
remove_loops	A logical scalar: should loop edges be removed?

Value

An **igraph** graph object.

Note

Experimental.

The **igraph** and **relations** packages need to be installed on your machine.

Author(s)

N. Frerebeau

See Also

Other graph tools: graph_create()

Examples

```
if (requireNamespace("igraph", quietly = TRUE) &&
    requireNamespace("relations", quietly = TRUE)) {
  ## Seven intervals
  int <- intervals(
    start = c(1, 2, 3, 6, 9, 13, 17),
    end = c(7, 4, 15, 14, 11, 18, 19),
    calendar = CE(),
```

```
    names = c("A", "B", "C", "D", "E", "F", "G")
  )

  ## Interval graph
  g <- graph_create(int, type = "interval")
  plot(g)

  ## Stratigraphic graph
  g <- graph_create(int, type = "strati")
  g <- graph_prune(g) # Remove redundant relations
  plot(g, layout = igraph::layout_with_sugiyama)
}
```

`gregorian` *Gregorian Calendar*

Description

Gregorian Calendar

Usage

```
BP(...)

b2k(...)

BC(...)

BCE(...)

AD(...)

CE(...)
```

Arguments

`...`	Currently not used.

Value

A `GregorianCalendar` object.

Author(s)

N. Frerebeau

See Also

calendar()

Other calendar tools: calendar(), calendar_get, convert(), get_calendar(), is_calendar(), julian

Examples

```
## Define time scales
calendar("BP")
calendar("AD")
calendar("julian")

## Shortcuts
BP()
AD()
J()
```

---

image *Heat Map*

Description

Heat Map

Usage

```
## S4 method for signature 'TimeSeries'
image(x, calendar = get_calendar(), k = 1, ...)
```

Arguments

x	A TimeSeries object.
calendar	A TimeScale object specifying the target calendar (see calendar()).
k	An integer specifying the slice of x along the third dimension to be plotted.
...	Further parameters to be passed to graphics::image().

Value

image() is called for its side-effects: it results in a graphic being displayed. Invisibly returns x.

Author(s)

N. Frerebeau

See Also

graphics::image()

Other plotting tools: plot(), year_axis()

Examples

```
## Create 6 time-series of 50 observations
## Sampled every two years starting from 2000 BP
X <- series(
  object = matrix(rnorm(300), nrow = 50, ncol = 6),
  time = seq(2000, by = -2, length.out = 50),
  calendar = BP()
)

## Image
image(X, calendar = CE())
```

---

intervals *Create Time Intervals*

Description

An Interval is elapsed time in seconds between two specific years.

Usage

```
intervals(start, end, calendar, ...)

## S4 method for signature 'RataDie,RataDie,missing'
intervals(start, end, names = NULL)

## S4 method for signature 'numeric,numeric,TimeScale'
intervals(start, end, calendar, scale = 1, names = NULL)
```

Arguments

start	A numeric vector of (decimal) years or a RataDie object (see fixed()) giving the beginning of the time intervals.
end	A numeric vector of (decimal) years or a RataDie object (see fixed()) giving the end of the time intervals.
calendar	A TimeScale object specifying the calendar of time (see calendar()). If missing, time must be a RataDie object.
...	Currently not used.
names	A character string specifying the names of the time series.
scale	A length-one numeric vector specifying the number of years represented by one unit. It should be a power of 10 (i.e. 1000 means ka).

Value

A TimeIntervals object.

Author(s)

N. Frerebeau

Examples

```
## Create time intervals
int <- intervals(
  start = c(625, 700, 1200, 1225, 1250, 500, 1000, 1200,
            1325, 1375, 1200, 1300, 1375, 1275, 1325),
  end = c(750, 825, 1250, 1275, 1325, 700, 1300, 1325,
          1400, 1500, 1300, 1375, 1500, 1325, 1425),
  calendar = CE()
)

## Plot intervals
plot(int) # Default calendar

## Overlap
overlap(int, calendar = CE())
```

---

is_calendar *Is an Object a Calendar?*

Description

Test inheritance relationships between an object and a calendar class.

Usage

```
is_calendar(object)

is_gregorian(object)

is_julian(object)
```

Arguments

object	Any R object.

Value

A logical scalar.

Author(s)

N. Frerebeau

See Also

Other calendar tools: calendar(), calendar_get, convert(), get_calendar(), gregorian, julian

---

julian *Julian Calendar*

Description

Julian Calendar

Usage

```
J(...)
```

Arguments

...	Currently not used.

Value

A JulianCalendar object.

Author(s)

N. Frerebeau

See Also

calendar()

Other calendar tools: calendar(), calendar_get, convert(), get_calendar(), gregorian, is_calendar()

Examples

```
## Define time scales
calendar("BP")
calendar("AD")
calendar("julian")

## Shortcuts
BP()
AD()
J()
```

labels *Labels*

Description

Find a suitable set of labels from an object.

Usage

```
## S4 method for signature 'TimeSeries'
labels(object, ...)

## S4 method for signature 'TimeIntervals'
labels(object, ...)
```

Arguments

object	An R object.
...	Currently not used.

Value

A character vector.

Author(s)

N. Frerebeau

See Also

Other mutators: flip(), length, names

length *Length*

Description

Get the length of an object.

Usage

```
## S4 method for signature 'TimeIntervals'
length(x)
```

Arguments

x	An R object.

Value

A length-one integer vector.

Author(s)

N. Frerebeau

See Also

Other mutators: flip(), labels(), names

---

names *Names*

Description

Get or set the names of an object.

Usage

```
## S4 method for signature 'TimeIntervals'
names(x)

## S4 replacement method for signature 'TimeIntervals'
names(x) <- value
```

Arguments

x	An R object.
value	A character vector of up to the same length as x, or NULL.

Value

The updated object.

Author(s)

N. Frerebeau

See Also

Other mutators: flip(), labels(), length

overlap *Time Overlap*

Description

Computes the length of overlap of time intervals.

Usage

```
overlap(x, ...)

## S4 method for signature 'TimeIntervals'
overlap(x, calendar = NULL, aggregate = TRUE)
```

Arguments

x	A TimeIntervals object.
...	Currently not used.
calendar	A TimeScale object specifying the target calendar (see calendar()). If NULL (the default), *rata die* are returned.
aggregate	A logical scalar: should disjoint intervals referring to the same event be aggregated?

Details

The overlap of two time intervals is a difference between the minimum value of the two upper boundaries and the maximum value of the two lower boundaries, plus 1.

Value

A symmetric numeric matrix of years.

Author(s)

N. Frerebeau

See Also

Other temporal relations: relations

Examples

```
## Create time intervals
int <- intervals(
  start = c(625, 700, 1200, 1225, 1250, 500, 1000, 1200,
            1325, 1375, 1200, 1300, 1375, 1275, 1325),
  end = c(750, 825, 1250, 1275, 1325, 700, 1300, 1325,
          1400, 1500, 1300, 1375, 1500, 1325, 1425),
```

```
  calendar = CE()
)

## Plot intervals
plot(int) # Default calendar

## Overlap
overlap(int, calendar = CE())
```

---

plot *Plot Time Series and Time Intervals*

Description

Plot Time Series and Time Intervals

Usage

```
## S4 method for signature 'TimeIntervals,missing'
plot(
  x,
  calendar = get_calendar(),
  groups = NULL,
  sort = TRUE,
  decreasing = FALSE,
  xlab = NULL,
  ylab = NULL,
  main = NULL,
  sub = NULL,
  ann = graphics::par("ann"),
  axes = TRUE,
  frame.plot = axes,
  panel.first = NULL,
  panel.last = NULL,
  ...
)

## S4 method for signature 'TimeSeries,missing'
plot(
  x,
  facet = c("multiple", "single"),
  calendar = get_calendar(),
  panel = graphics::lines,
  flip = FALSE,
  ncol = NULL,
  xlab = NULL,
  ylab = NULL,
  main = NULL,
```

```
  sub = NULL,
  ann = graphics::par("ann"),
  axes = TRUE,
  frame.plot = axes,
  panel.first = NULL,
  panel.last = NULL,
  ...
)
```

Arguments

x	A TimeSeries or a TimeIntervals object.
calendar	A TimeScale object specifying the target calendar (see calendar()).
groups	A character vector specifying the group each interval belongs to.
sort	A logical scalar: should the data be sorted in chronological order?
decreasing	A logical scalar: should the sort order be decreasing? Only used if sort is TRUE.
xlab, ylab	A character vector giving the x and y axis labels.
main	A character string giving a main title for the plot.
sub	A character string giving a subtitle for the plot.
ann	A logical scalar: should the default annotation (title and x and y axis labels) appear on the plot?
axes	A logical scalar: should axes be drawn on the plot?
frame.plot	A logical scalar: should a box be drawn around the plot?
panel.first	An expression to be evaluated after the plot axes are set up but before any plotting takes place. This can be useful for drawing background grids.
panel.last	An expression to be evaluated after plotting has taken place but before the axes, title and box are added.
...	Further parameters to be passed to panel (e.g. graphical parameters).
facet	A character string specifying whether the series should be plotted separately (with a common time axis) or on a single plot? It must be one of "multiple" or "single". Any unambiguous substring can be given.
panel	A function in the form function(x, y, ...) which gives the action to be carried out in each panel of the display. The default is graphics::lines().
flip	A logical scalar: should the y-axis (ticks and numbering) be flipped from side 2 (left) to 4 (right) from series to series when facet is "multiple"?
ncol	An integer specifying the number of columns to use when facet is "multiple". Defaults to 1 for up to 4 series, otherwise to 2.

Value

plot() is called for its side-effects: it results in a graphic being displayed. Invisibly returns x.

Author(s)

N. Frerebeau

See Also

graphics::plot()

Other plotting tools: image(), year_axis()

Examples

```
## Create 6 time-series of 50 observations
## Sampled every two years starting from 2000 BP
X <- series(
  object = matrix(rnorm(300), nrow = 50, ncol = 6),
  time = seq(2000, by = -2, length.out = 50),
  calendar = BP()
)

## Multiple
plot(X) # Default calendar
plot(X, calendar = BP(), flip = TRUE) # BP
plot(X, calendar = b2k(), ncol = 1) # b2k

## Single
plot(X, facet = "single") # CE
plot(X, facet = "single", calendar = BP()) # BP

## Create 6 x 3 time-series of 50 observations
## Sampled every two years starting from 2000 BP
X <- series(
  object = array(rnorm(900), dim = c(50, 6, 3)),
  time = seq(2000, by = 2, length.out = 50),
  calendar = BP()
)
plot(X, calendar = BP(), flip = TRUE) # BP
plot(X, calendar = b2k(), ncol = 1) # b2k

## Graphical parameters
plot(X, lwd = c(1, 2, 3), col = c("#004488", "#DDAA33", "#BB5566"))
plot(X, type = "b", pch = 16, col = c("#004488", "#DDAA33", "#BB5566"))
plot(X, type = "p", pch = c(16, 17, 18), cex = c(1, 2, 3))
```

---

pretty *Pretty Breakpoints*

Description

Pretty Breakpoints

Usage

```
## S4 method for signature 'RataDie'
pretty(x, calendar = get_calendar(), ...)
```

Arguments

x	A RataDie object.
calendar	A TimeScale object specifying the target calendar (see calendar()).
...	Further parameters to be passed to base::pretty().

Details

pretty() computes a vector of increasing numbers which are "pretty" in decimal notation of calendar. Pretty breakpoints are then converted to *rata die*.

Value

A RataDie object.

See Also

Other fixed date tools: arithmetic, as_date(), as_decimal(), as_fixed(), as_year(), fixed(), fixed_gregorian, fixed_julian, format()

RataDie-class *RataDie*

Description

An S4 class to represent a vector of *rata die*.

Details

Rata die (fixed date) are represented as the number of days since 01-01-01 (Gregorian), with negative values for earlier dates.

It is intended that the date should be an integer value, but this is not enforced in the internal representation.

Slots

.Data A numeric vector giving the *rata die* values.

Note

This class inherits from numeric.

Author(s)

N. Frerebeau

See Also

Other classes: GregorianCalendar-class, JulianCalendar-class, TimeIntervals-class, TimeScale-class, TimeSeries-class

Other time classes: TimeIntervals-class, TimeSeries-class

---

relations *Temporal Relations*

Description

Test for the logical relation between time intervals according to Allen's typology.

Usage

```
precedes(x, ...)

preceded_by(x, ...)

meets(x, ...)

met_by(x, ...)

overlaps(x, ...)

overlapped_by(x, ...)

finishes(x, ...)

finished_by(x, ...)

contains(x, ...)

during(x, ...)

starts(x, ...)

started_by(x, ...)

equals(x, ...)

## S4 method for signature 'TimeIntervals'
precedes(x, ...)
```

```
## S4 method for signature 'TimeIntervals'
preceded_by(x, ...)

## S4 method for signature 'TimeIntervals'
meets(x, ...)

## S4 method for signature 'TimeIntervals'
met_by(x, ...)

## S4 method for signature 'TimeIntervals'
overlaps(x, ...)

## S4 method for signature 'TimeIntervals'
overlapped_by(x, ...)

## S4 method for signature 'TimeIntervals'
finishes(x, ...)

## S4 method for signature 'TimeIntervals'
finished_by(x, ...)

## S4 method for signature 'TimeIntervals'
contains(x, ...)

## S4 method for signature 'TimeIntervals'
during(x, ...)

## S4 method for signature 'TimeIntervals'
starts(x, ...)

## S4 method for signature 'TimeIntervals'
started_by(x, ...)

## S4 method for signature 'TimeIntervals'
equals(x, ...)
```

Arguments

x	A TimeIntervals object.
...	Currently not used.

Details

Allen (1983) proposed thirteen basic relations between time intervals that are (Alspaugh 2019):

- **Distinct**: no pair of definite intervals can be related by more than one of the relationships.
- **Exhaustive:** any pair of definite intervals are described by one of the relations.
- **Qualitative:** no numeric time spans are considered.

Relation			**Converse**
precedes	(p)	(P)	preceded by
meets	(m)	(M)	met by
overlaps	(o)	(O)	overlapped by
finished by	(F)	(f)	finishes
contains	(D)	(d)	during
starts	(s)	(S)	started by
equals	(e)		

A *precedes* B

```
A ===
B       ===
```

A *preceded by* B

```
A       ===
B ===
```

A *meets* B

```
A ===
B      ===
```

A *met by* B

```
A ===
B      ===
```

A *overlaps* B

```
A ===
B    ===
```

A *overlapped by* B

```
A    ===
B ===
```

A *finished by* B

```
A =====
B    ===
```

A *finishes* B

```
A    ===
B =====
```

A *contains* B

```
A =====
B   ===
```

A *during* B

```
A   ===
B =====
```

A *starts* B

```
A ===
B =====
```

A *started by* B

```
A =====
B ===
```

A *equals* B

```
A ===
B ===
```

Value

A two-columns `matrix` where each row specifies one relation.

Author(s)

N. Frerebeau

References

Allen, J. F. (1983). Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26(11): 832-843. doi:10.1145/182.358434.

Alspaugh, T. (2019). Allen's Interval Algebra. URL: https://thomasalspaugh.org/pub/fnd/allen.html.

See Also

Other temporal relations: `overlap()`

Examples

```
## Seven intervals
int <- intervals(
  start = c(1, 2, 3, 6, 9, 13, 17),
  end = c(7, 4, 15, 14, 11, 18, 19),
  calendar = CE(),
  names = c("A", "B", "C", "D", "E", "F", "G")
)

## Plot intervals
plot(int)

## Temporal relations
precedes(int) # A precedes E...

overlaps(int) # A overlaps C...

contains(int) # A contains B...
```

---

series *Create Time Series*

Description

Create Time Series

Usage

```
series(object, time, calendar, ...)

## S4 method for signature 'array,RataDie,missing'
series(object, time, names = NULL)

## S4 method for signature 'array,numeric,TimeScale'
series(object, time, calendar, scale = 1, names = NULL)

## S4 method for signature 'matrix,numeric,TimeScale'
series(object, time, calendar, scale = 1, names = NULL)

## S4 method for signature 'matrix,RataDie,missing'
series(object, time, names = NULL)

## S4 method for signature 'numeric,numeric,TimeScale'
series(object, time, calendar, scale = 1, names = NULL)

## S4 method for signature 'numeric,RataDie,missing'
series(object, time, names = NULL)
```

```
## S4 method for signature 'data.frame,numeric,TimeScale'
series(object, time, calendar, scale = 1, names = NULL)

## S4 method for signature 'data.frame,RataDie,missing'
series(object, time, names = NULL)
```

Arguments

object	A numeric vector, matrix or array of the observed time-series values. A data.frame will be coerced to a numeric matrix via data.matrix().
time	A numeric vector of (decimal) years or a RataDie object (see fixed()).
calendar	A TimeScale object specifying the calendar of time (see calendar()). If missing, time must be a RataDie object.
...	Currently not used.
names	A character string specifying the names of the time series.
scale	A length-one numeric vector specifying the number of years represented by one unit. It should be a power of 10 (i.e. 1000 means ka).

Details

Data will be sorted in chronological order.

Value

A TimeSeries object.

Author(s)

N. Frerebeau

Examples

```
## Create time-series of 20 observations

## Univariate
## Sampled every years starting from 1029 BCE
(X <- series(rnorm(30), time = 1029:1000, calendar = BCE()))

## Terminal and sampling times (returns rata die)
start(X)
end(X)
time(X)
span(X)

## Multivariate
## Sampled every century starting from 1000 CE
(Y <- series(matrix(rnorm(90), 30, 3), time = 1000:1029, calendar = CE()))

## Terminal and sampling times (returns Gregorian Common Era years)
start(Y, calendar = CE())
```

```
end(Y, calendar = CE())
time(Y, calendar = CE())
span(Y, calendar = CE())

## Coerce to data frame
df <- as.data.frame(Y, calendar = BP())
head(df)
```

---

span *Duration*

Description

Get the duration of time series or intervals.

Usage

```
span(x, ...)

## S4 method for signature 'TimeSeries'
span(x, calendar = NULL)

## S4 method for signature 'TimeIntervals'
span(x, calendar = NULL)
```

Arguments

x	A TimeSeries or a TimeIntervals object.
...	Currently not used.
calendar	A TimeScale object specifying the target calendar (see calendar()). If NULL (the default), *rata die* are returned.

Value

A numeric vector of years.

Author(s)

N. Frerebeau

See Also

Other tools: start(), time(), window()

Examples

```
## Create time intervals
int <- intervals(
  start = c(625, 700, 1200, 1225, 1250, 500, 1000, 1200,
            1325, 1375, 1200, 1300, 1375, 1275, 1325),
  end = c(750, 825, 1250, 1275, 1325, 700, 1300, 1325,
          1400, 1500, 1300, 1375, 1500, 1325, 1425),
  calendar = CE()
)

## Get time durations
span(int, calendar = CE())
```

`start` *Terminal Times*

Description

Get the times the first and last observations were taken.

Usage

```
## S4 method for signature 'TimeSeries'
start(x, calendar = NULL, ...)

## S4 method for signature 'TimeIntervals'
start(x, calendar = NULL, ...)

## S4 method for signature 'TimeSeries'
end(x, calendar = NULL, ...)

## S4 method for signature 'TimeIntervals'
end(x, calendar = NULL, ...)
```

Arguments

`x`	A `TimeSeries` object.
`calendar`	A `TimeScale` object specifying the target calendar (see `calendar()`). If `NULL` (the default), *rata die* are returned.
`...`	Currently not used.

Value

A `numeric` vector of decimal years (if `calendar` is not `NULL`).

Author(s)

N. Frerebeau

See Also

Other tools: span(), time(), window()

Examples

```
## Create time-series of 20 observations

## Univariate
## Sampled every years starting from 1029 BCE
(X <- series(rnorm(30), time = 1029:1000, calendar = BCE()))

## Terminal and sampling times (returns rata die)
start(X)
end(X)
time(X)
span(X)

## Multivariate
## Sampled every century starting from 1000 CE
(Y <- series(matrix(rnorm(90), 30, 3), time = 1000:1029, calendar = CE()))

## Terminal and sampling times (returns Gregorian Common Era years)
start(Y, calendar = CE())
end(Y, calendar = CE())
time(Y, calendar = CE())
span(Y, calendar = CE())

## Coerce to data frame
df <- as.data.frame(Y, calendar = BP())
head(df)
```

---

subset *Extract or Replace Parts of an Object*

Description

Operators acting on objects to extract or replace parts.

Usage

```
## S4 method for signature 'RataDie'
x[i]

## S4 method for signature 'TimeIntervals'
x[i]

## S4 method for signature 'TimeSeries'
x[i, j, k, drop = FALSE]
```

Arguments

x	An object from which to extract element(s) or in which to replace element(s).
i, j, k	Indices specifying elements to extract or replace.
drop	A logical scalar: should the result be coerced to the lowest possible dimension? This only works for extracting elements, not for the replacement.

Value

A subsetted object.

Author(s)

N. Frerebeau

---

time *Sampling Times*

Description

Get the sampling times:

- `time()` creates the vector of times at which a time series was sampled.
- `frequency()` returns the mean number of samples per unit time.

Usage

```
## S4 method for signature 'TimeSeries'
time(x, calendar = NULL, ...)

## S4 method for signature 'TimeSeries'
frequency(x, ...)
```

Arguments

x	A TimeSeries object.
calendar	A TimeScale object specifying the target calendar (see calendar()). If NULL (the default), *rata die* are returned.
...	Currently not used.

Value

A numeric vector of decimal years (if calendar is not NULL).

Author(s)

N. Frerebeau

See Also

Other tools: span(), start(), window()

Examples

```
## Create time-series of 20 observations

## Univariate
## Sampled every years starting from 1029 BCE
(X <- series(rnorm(30), time = 1029:1000, calendar = BCE()))

## Terminal and sampling times (returns rata die)
start(X)
end(X)
time(X)
span(X)

## Multivariate
## Sampled every century starting from 1000 CE
(Y <- series(matrix(rnorm(90), 30, 3), time = 1000:1029, calendar = CE()))

## Terminal and sampling times (returns Gregorian Common Era years)
start(Y, calendar = CE())
end(Y, calendar = CE())
time(Y, calendar = CE())
span(Y, calendar = CE())

## Coerce to data frame
df <- as.data.frame(Y, calendar = BP())
head(df)
```

TimeIntervals-class *TimeIntervals*

Description

An S4 class to represent time intervals.

Slots

`.Id` A character vector specifying the identifier/name of intervals. Duplicated values are interpreted as disjoint intervals referring to the same event.

`.Start` A RataDie object giving the start time of the intervals.

`.End` A RataDie object giving the end time of the intervals.

Author(s)

N. Frerebeau

See Also

Other classes: GregorianCalendar-class, JulianCalendar-class, RataDie-class, TimeScale-class, TimeSeries-class

Other time classes: RataDie-class, TimeSeries-class

TimeScale-class *TimeScale*

Description

A virtual S4 class to represent a calendar.

Slots

label A character string specifying the abbreviated label of the time scale.

name A character string specifying the name of the time scale.

epoch A numeric value specifying the epoch year from which years are counted (starting date of the calendar, in years). Allows to define multiple era of a calendar.

fixed A numeric value specifying the reference date of the calendar (in *rata die*).

direction An integer specifying if years are counted backwards (-1) or forwards (1) from epoch.

year A numeric value giving the average length of the year in solar days.

Author(s)

N. Frerebeau

See Also

Other classes: GregorianCalendar-class, JulianCalendar-class, RataDie-class, TimeIntervals-class, TimeSeries-class

Other calendar classes: GregorianCalendar-class, JulianCalendar-class

TimeSeries-class *TimeSeries*

Description

An S4 class to represent time series.

Details

A time series object is an $n \times m \times p$ array, with n being the number of observations, m being the number of series and with the p columns of the third dimension containing extra variables for each series.

Slots

`.Data` A $n \times m \times p$ `numeric` `array` giving the observed time-series values.

`.Time` A length-n `RataDie` object.

Note

This class inherits from `array`.

Author(s)

N. Frerebeau

See Also

Other classes: `GregorianCalendar-class`, `JulianCalendar-class`, `RataDie-class`, `TimeIntervals-class`, `TimeScale-class`

Other time classes: `RataDie-class`, `TimeIntervals-class`

window *Time Windows*

Description

Extracts the subset of the object `x` observed between the times `start` and `end` (expressed in *rata die*).

Usage

```
## S4 method for signature 'TimeSeries'
window(x, start = NULL, end = NULL, ...)
```

Arguments

x	A TimeSeries object.
start	A length-one numeric vector specifying the start time of the period of interest.
end	A length-one numeric vector specifying the end time of the period of interest.
...	Currently not used.

Value

A TimeSeries object.

Author(s)

N. Frerebeau

See Also

Other tools: span(), start(), time()

Examples

```
## Create 3 time-series of 100 observations
## Sampled every years starting from 1000 CE
(x <- series(matrix(rnorm(300), 100, 3), time = 1000:1099, calendar = CE()))

## Subset between 1025 and 1050 CE
(y <- window(x, start = 374009, end = 383140))
```

---

year_axis *Time Series Plotting Functions*

Description

Time Series Plotting Functions

Usage

```
year_axis(
  side,
  at = NULL,
  format = c("a", "ka", "Ma", "Ga"),
  labels = TRUE,
  calendar = get_calendar("current"),
  current_calendar = get_calendar("current"),
  ...
)
```

Arguments

side	An integer specifying which side of the plot the axis is to be drawn on. The axis is placed as follows: 1=below, 2=left, 3=above and 4=right.
at	A numeric vector giving the points at which tick-marks are to be drawn. If NULL, tickmark locations are computed.
format	A character string specifying the prefix. It should be one of "a", "ka", "Ma" or "Ga". If TRUE, a good guess for an appropriate format is made.
labels	A logical scalar specifying whether annotations are to be made at the tick-marks, or a vector of character strings to be placed at the tickpoints.
calendar	A TimeScale object specifying the target calendar (see calendar()).
current_calendar	A TimeScale object specifying the calendar used by the last call to plot().
...	Further parameters to be passed to graphics::axis(). (e.g. graphical parameters).

Value

year_axis() is called it for its side-effects.

Author(s)

N. Frerebeau

See Also

Other plotting tools: image(), plot()

Examples

```
## Create a time-series of 300 observations
## Sampled every two years starting from 2000 BP
X <- series(
  object = rnorm(300),
  time = seq(2000, by = -2, length.out = 300),
  calendar = BP()
)

## Axis
plot(X, axes = FALSE, calendar = BP()) # Remove axes
year_axis(side = 1) # Same calendar as last plot
year_axis(side = 3, calendar = CE()) # Specific calendar
mtext(format(CE()), side = 3, line = 3)

## Grid
plot(X, panel.first = graphics::grid())
```

Index

∗ **calendar classes**
TimeScale-class, 50
∗ **calendar tools**
calendar, 10
calendar_get, 12
convert, 14
get_calendar, 22
gregorian, 27
is_calendar, 30
julian, 31
∗ **classes**
RataDie-class, 38
TimeIntervals-class, 49
TimeScale-class, 50
TimeSeries-class, 51
∗ **coercion methods**
as.data.frame, 4
∗ **datasets**
dates, 15
∗ **fixed date tools**
arithmetic, 3
as_date, 6
as_decimal, 7
as_fixed, 8
as_year, 9
fixed, 15
fixed_gregorian, 17
fixed_julian, 19
format, 21
pretty, 37
∗ **graph tools**
graph_create, 23
graph_prune, 25
∗ **mutators**
flip, 20
labels, 32
length, 32
names, 33
∗ **plotting tools**
image, 28
plot, 35
year_axis, 52
∗ **subsetting methods**
subset, 47
∗ **temporal relations**
overlap, 34
relations, 39
∗ **time classes**
RataDie-class, 38
TimeIntervals-class, 49
TimeSeries-class, 51
∗ **time intervals**
intervals, 29
∗ **time series**
series, 43
∗ **tools**
span, 45
start, 46
time, 48
window, 51
.RataDie *(RataDie-class)*, 38
.TimeIntervals *(TimeIntervals-class)*, 49
.TimeScale *(TimeScale-class)*, 50
.TimeSeries *(TimeSeries-class)*, 51
[,RataDie-method *(subset)*, 47
[,TimeIntervals-method *(subset)*, 47
[,TimeSeries-method *(subset)*, 47

AD *(gregorian)*, 27
Arith,numeric,RataDie-method *(arithmetic)*, 3
Arith,RataDie,numeric-method *(arithmetic)*, 3
Arith,RataDie,RataDie-method *(arithmetic)*, 3
arithmetic, 3, *6, 7*, *9, 10*, *16*, *18, 19*, *22*, *38*
array, *51*
as.data.frame, 4

as.data.frame,TimeIntervals-method (as.data.frame), 4
as.data.frame,TimeSeries-method (as.data.frame), 4
as_date, 6
as_date(), *4*, *7*, *9, 10*, *16*, *18, 19*, *22*, *38*
as_date,numeric,GregorianCalendar-method (as_date), 6
as_date,numeric,JulianCalendar-method (as_date), 6
as_date-method (as_date), 6
as_decimal, 7
as_decimal(), *4*, *6*, *9, 10*, *16*, *18, 19*, *22*, *38*
as_decimal,numeric,numeric,numeric,GregorianCalendar-method (as_decimal), 7
as_decimal,numeric,numeric,numeric,JulianCalendar-method (as_decimal), 7
as_decimal-method (as_decimal), 7
as_fixed, 8
as_fixed(), *4*, *6, 7*, *10*, *16*, *18, 19*, *22*, *38*
as_fixed,numeric-method (as_fixed), 8
as_fixed-method (as_fixed), 8
as_year, 9
as_year(), *4*, *6, 7*, *9*, *16*, *18, 19*, *22*, *38*
as_year,numeric,GregorianCalendar-method (as_year), 9
as_year,numeric,JulianCalendar-method (as_year), 9
as_year-method (as_year), 9

b2k (gregorian), 27
base::pretty(), *38*
BC (gregorian), 27
BCE (gregorian), 27
BP (gregorian), 27

calendar, 10
calendar(), *4*, *6, 7*, *9*, *13, 14*, *16*, *21*, *23*, *28, 29*, *31*, *34*, *36*, *38*, *44–46*, *48*, *53*
calendar,character-method (calendar), 10
calendar-method (calendar), 10
calendar_direction (calendar_get), 12
calendar_direction,NULL-method (calendar_get), 12
calendar_direction,TimeScale-method (calendar_get), 12
calendar_direction-method (calendar_get), 12
calendar_epoch (calendar_get), 12
calendar_epoch,TimeScale-method (calendar_get), 12
calendar_epoch-method (calendar_get), 12
calendar_fixed (calendar_get), 12
calendar_fixed,TimeScale-method (calendar_get), 12
calendar_fixed-method (calendar_get), 12
calendar_get, *11*, 12, *14*, *23*, *28*, *31*
calendar_label (calendar_get), 12
calendar_label,TimeScale-method (calendar_get), 12
calendar_label-method (calendar_get), 12
calendar_name (calendar_get), 12
calendar_name,TimeScale-method (calendar_get), 12
calendar_name-method (calendar_get), 12
calendar_unit (calendar_get), 12
calendar_unit,TimeScale-method (calendar_get), 12
calendar_unit-method (calendar_get), 12
calendar_year (calendar_get), 12
calendar_year,TimeScale-method (calendar_get), 12
calendar_year-method (calendar_get), 12
CE (gregorian), 27
character, *11*, *13*, *21–24*, *29*, *32, 33*, *36*, *44*, *49, 50*, *53*
contains (relations), 39
contains,TimeIntervals-method (relations), 39
contains-method (relations), 39
convert, 14
convert(), *11*, *13*, *23*, *28*, *31*
convert,character,character-method (convert), 14
convert,TimeScale,TimeScale-method (convert), 14
convert-method (convert), 14

data.frame, *4, 5*, *15*, *44*
data.frame(), *4*
data.matrix(), *44*
dates, 15
during (relations), 39
during,TimeIntervals,missing-method (relations), 39
during,TimeIntervals-method (relations), 39
during-method (relations), 39

end,TimeIntervals-method *(start)*, 46
end,TimeSeries-method *(start)*, 46
end-method *(start)*, 46
equals *(relations)*, 39
equals,TimeIntervals,missing-method *(relations)*, 39
equals,TimeIntervals-method *(relations)*, 39
equals-method *(relations)*, 39
era::era(), *11*

finished_by *(relations)*, 39
finished_by,TimeIntervals-method *(relations)*, 39
finished_by-method *(relations)*, 39
finishes *(relations)*, 39
finishes,TimeIntervals,missing-method *(relations)*, 39
finishes,TimeIntervals-method *(relations)*, 39
finishes-method *(relations)*, 39
fixed, 15
fixed(), *4*, *6, 7*, *9, 10*, *18, 19*, *22*, *29*, *38*, *44*
fixed,numeric,missing,missing,GregorianCalendar-method *(fixed)*, 15
fixed,numeric,missing,missing,JulianCalendar-method *(fixed)*, 15
fixed,numeric,numeric,numeric,GregorianCalendar-method *(fixed)*, 15
fixed,numeric,numeric,numeric,JulianCalendar-method *(fixed)*, 15
fixed-method *(fixed)*, 15
fixed_from_AD *(fixed_gregorian)*, 17
fixed_from_b2k *(fixed_gregorian)*, 17
fixed_from_BC *(fixed_gregorian)*, 17
fixed_from_BCE *(fixed_gregorian)*, 17
fixed_from_BP *(fixed_gregorian)*, 17
fixed_from_CE *(fixed_gregorian)*, 17
fixed_from_julian *(fixed_julian)*, 19
fixed_gregorian, *4*, *6, 7*, *9, 10*, *16*, 17, *19*, *22*, *38*
fixed_julian, *4*, *6, 7*, *9, 10*, *16*, *18*, 19, *22*, *38*
fixed_to_AD *(fixed_gregorian)*, 17
fixed_to_b2k *(fixed_gregorian)*, 17
fixed_to_BC *(fixed_gregorian)*, 17
fixed_to_BCE *(fixed_gregorian)*, 17
fixed_to_BP *(fixed_gregorian)*, 17
fixed_to_CE *(fixed_gregorian)*, 17
fixed_to_julian *(fixed_julian)*, 19

flip, 20
flip(), *32, 33*
flip,TimeSeries-method *(flip)*, 20
format, 21
format(), *4*, *6, 7*, *9, 10*, *16*, *18, 19*, *38*
format,RataDie-method *(format)*, 21
format,TimeIntervals-method *(format)*, 21
format,TimeScale-method *(format)*, 21
frequency,TimeSeries-method *(time)*, 48
frequency-method *(time)*, 48
function, *14*, *36*

get_calendar, 22
get_calendar(), *11*, *13, 14*, *28*, *31*
graph_create, 23
graph_create(), *26*
graph_create,data.frame-method *(graph_create)*, 23
graph_create,matrix-method *(graph_create)*, 23
graph_create,TimeIntervals-method *(graph_create)*, 23
graph_create-method *(graph_create)*, 23
graph_prune, 25
graph_prune(), *25*
graph_prune,igraph-method *(graph_prune)*, 25
graph_prune-method *(graph_prune)*, 25
graphical parameters, *36*, *53*
graphics::axis(), *53*
graphics::image(), *28*
graphics::lines(), *36*
graphics::plot(), *37*
gregorian, *11*, *13, 14*, *23*, 27, *31*
GregorianCalendar, *27*

image, 28
image(), *23*, *37*, *53*
image,TimeSeries-method *(image)*, 28
integer, *13*, *16*, *28*, *33*, *36*, *50*, *53*
intervals, 29
intervals,numeric,numeric,TimeScale-method *(intervals)*, 29
intervals,RataDie,RataDie,missing-method *(intervals)*, 29
intervals-method *(intervals)*, 29
is_calendar, 30
is_calendar(), *11*, *13, 14*, *23*, *28*, *31*
is_gregorian *(is_calendar)*, 30

is_julian (is_calendar), 30

J (julian), 31
julian, *11*, *13, 14*, *23*, *28*, 31, *31*
JulianCalendar, *31*

labels, 32
labels(), *20*, *33*
labels,TimeIntervals-method (labels), 32
labels,TimeSeries-method (labels), 32
length, *20*, 32, *32, 33*
length,TimeIntervals-method (length), 32
logical, *3*, *9*, *21*, *24*, *26*, *30*, *34*, *36*, *48*, *53*

matrix, *24*, *34*
meets (relations), 39
meets,TimeIntervals-method (relations), 39
meets-method (relations), 39
met_by (relations), 39
met_by,TimeIntervals,missing-method (relations), 39
met_by,TimeIntervals-method (relations), 39
met_by-method (relations), 39

names, *20*, *32*, 33, *33*
names,TimeIntervals-method (names), 33
names<-,TimeIntervals-method (names), 33
numeric, *3*, *6–8*, *10*, *13*, *16*, *18, 19*, *29*, *38*, *44–46*, *48*, *50*, *52, 53*

overlap, 34
overlap(), *42*
overlap,TimeIntervals-method (overlap), 34
overlap-method (overlap), 34
overlapped_by (relations), 39
overlapped_by,TimeIntervals,missing-method (relations), 39
overlapped_by,TimeIntervals-method (relations), 39
overlapped_by-method (relations), 39
overlaps (relations), 39
overlaps,TimeIntervals-method (relations), 39
overlaps-method (relations), 39

plot, 35
plot(), *23*, *28*, *53*
plot,TimeIntervals,missing-method (plot), 35
plot,TimeSeries,missing-method (plot), 35
preceded_by (relations), 39
preceded_by,TimeIntervals,missing-method (relations), 39
preceded_by,TimeIntervals-method (relations), 39
preceded_by-method (relations), 39
precedes (relations), 39
precedes,TimeIntervals-method (relations), 39
precedes-method (relations), 39
pretty, 37
pretty(), *4*, *6, 7*, *9, 10*, *16*, *18, 19*, *22*
pretty,RataDie-method (pretty), 37

RataDie, *3*, *6*, *8, 9*, *16*, *18, 19*, *21*, *29*, *38*, *44*, *49*, *51*
RataDie-class, 38
relations, *34*, 39

series, 43
series,array,numeric,TimeScale-method (series), 43
series,array,RataDie,missing-method (series), 43
series,data.frame,numeric,TimeScale-method (series), 43
series,data.frame,RataDie,missing-method (series), 43
series,matrix,numeric,TimeScale-method (series), 43
series,matrix,RataDie,missing-method (series), 43
series,numeric,numeric,TimeScale-method (series), 43
series,numeric,RataDie,missing-method (series), 43
series-method (series), 43
set_calendar (get_calendar), 22
sign, *13*
span, 45
span(), *47*, *49*, *52*
span,TimeIntervals-method (span), 45
span,TimeSeries-method (span), 45
span-method (span), 45
start, 46

start(), *45*, *49*, *52*
start,TimeIntervals-method *(*start*)*, 46
start,TimeSeries-method *(*start*)*, 46
start-method *(*start*)*, 46
started_by *(*relations*)*, 39
started_by,TimeIntervals,missing-method
*(*relations*)*, 39
started_by,TimeIntervals-method
*(*relations*)*, 39
started_by-method *(*relations*)*, 39
starts *(*relations*)*, 39
starts,TimeIntervals,missing-method
*(*relations*)*, 39
starts,TimeIntervals-method
*(*relations*)*, 39
starts-method *(*relations*)*, 39
subset, 47

time, 48
time(), *45*, *47*, *52*
time,TimeSeries-method *(*time*)*, 48
time-method *(*time*)*, 48
TimeIntervals, *4*, *24*, *29*, *34*, *36*, *40*, *45*
TimeIntervals-class, 49
TimeScale, *4*, *6, 7*, *9*, *11*, *13, 14*, *16*, *21*, *23*,
28, 29, *34*, *36*, *38*, *44–46*, *48*, *53*
TimeScale-class, 50
TimeSeries, *4*, *20*, *28*, *36*, *44–46*, *48*, *52*
TimeSeries-class, 51

window, 51
window(), *45*, *47*, *49*
window,TimeSeries-method *(*window*)*, 51
window-method *(*window*)*, 51

year_axis, 52
year_axis(), *28*, *37*