

# Package ‘aion’

July 22, 2025

**Title** Archaeological Time Series

**Version** 1.5.0

**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](https://doi.org/10.1017/9781107415058)>. This package 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/>

**BugReports** <https://codeberg.org/tesselle/aion/issues>

**Depends** R (>= 3.3)

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

**Suggests** folio (>= 1.5.0), knitr, markdown, rsvg, svglite,  
tinysnapshot, tinytest

**VignetteBuilder** knitr

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.3.2

**X-schema.org-applicationCategory** Archaeological Science

**X-schema.org-isPartOf** <https://www.tesselle.org>

**X-schema.org-keywords** time-series, r-package, archaeology,  
archaeological-science, chronology, 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' 'intervals.R' 'mutators.R' 'operators.R'

'overlap.R' 'plot.R' 'series.R' 'show.R' 'span.R' 'subset.R'  
 'time.R' 'validate.R' 'year.R' 'zzz.R'

**NeedsCompilation** no

**Author** Nicolas Frerebeau [aut, cre] (ORCID:

<<https://orcid.org/0000-0001-5759-4944>>),

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

Brice Lebrun [art] (ORCID: <<https://orcid.org/0000-0001-7503-8685>>),

Logo designer),

Université Bordeaux Montaigne [fnd] (ROR: <<https://ror.org/03pbgwk21>>),

CNRS [fnd] (ROR: <<https://ror.org/02feahw73>>)

**Repository** CRAN

**Date/Publication** 2025-04-29 16:10:02 UTC

## Contents

arithmetic . . . . .	3
as.data.frame . . . . .	4
as_date . . . . .	5
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
format . . . . .	20
get_calendar . . . . .	21
gregorian . . . . .	22
image . . . . .	23
intervals . . . . .	24
is_calendar . . . . .	25
julian . . . . .	26
labels . . . . .	27
length . . . . .	28
overlap . . . . .	28
plot . . . . .	29
pretty . . . . .	32
RataDie-class . . . . .	33
series . . . . .	34
span . . . . .	35
start . . . . .	37
subset . . . . .	38
time . . . . .	39

<i>arithmetic</i>	3
TimeIntervals-class . . . . .	40
TimeScale-class . . . . .	41
TimeSeries-class . . . . .	42
window . . . . .	42
year_axis . . . . .	43
<b>Index</b>	<b>45</b>

---

arithmetic	<i>Arithmetic Operators</i>
------------	-----------------------------

---

### 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 <a href="#">TimeSeries</a> or a <a href="#">TimeIntervals</a> object.
...	Further parameters to be passed to <a href="#">data.frame()</a> .
calendar	A <a href="#">TimeScale</a> object specifying the target calendar (see <a href="#">calendar()</a> ). If NULL (the default), <i>rata die</i> are returned.

**Value**

A [data.frame](#).

**Methods (by class)**

- `as.data.frame(TimeSeries)`: Returns a long [data.frame](#) with the following columns:
  - `time` The (decimal) years at which the time series was sampled.
  - `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, in (decimal) years.
  - `end` The end time of the intervals, in (decimal) years.

### Author(s)

N. Frerebeau

### See Also

Other mutators: [labels\(\)](#), [length\(\)](#), [subset\(\)](#)

### 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](https://doi.org/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	<i>Converts a Date to a Decimal of its Year</i>
------------	---

---

## 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 <a href="#">numeric</a> vector of years. If month and day are missing, decimal years are expected.
month	A <a href="#">numeric</a> vector of months.
day	A <a href="#">numeric</a> vector of days.
calendar	A <a href="#">TimeScale</a> object specifying the calendar of year, month and day (see <a href="#">calendar()</a> ).

## 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](https://doi.org/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	<i>Year Conversion from Rata Die</i>
---------	--------------------------------------

---

**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 <a href="#">RataDie</a> object (see <a href="#">fixed()</a> ).
calendar	A <a href="#">TimeScale</a> object specifying the target calendar (see <a href="#">calendar()</a> ).
...	Currently not used.
decimal	A <a href="#">logical</a> 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:

<b>label</b>	<b>era</b>	<b>calendar</b>
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**

<code>from</code>	A <a href="#">TimeScale</a> object describing the source calendar.
<code>to</code>	A <a href="#">TimeScale</a> object describing the target calendar.
<code>...</code>	Currently not used.

**Value**

A [function](#) that when called with a single numeric argument (fractional 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 different 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](https://doi.org/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 <a href="#">numeric</a> vector of years. If month and day are missing, decimal years are expected.
month	A <a href="#">numeric</a> vector of months.
day	A <a href="#">numeric</a> vector of days.
calendar	A <a href="#">TimeScale</a> object specifying the calendar of year, month and day (see <a href="#">calendar()</a> ).
...	Currently not used.
scale	A length-one <a href="#">integer</a> 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 <b>numeric</b> vector of years. If month and day are missing, decimal years are expected.
month	A <b>numeric</b> vector of months.
day	A <b>numeric</b> vector of days.
object	A <b>RataDie</b> object (see <code>fixed()</code> ).

### 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 <b>numeric</b> vector of years. If month and day are missing, decimal years are expected.
month	A <b>numeric</b> vector of months.
day	A <b>numeric</b> vector of days.
object	A <b>RataDie</b> object (see <code>fixed()</code> ).

## 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)
```

---

format	<i>Date Conversion to Character</i>
--------	-------------------------------------

---

**Description**

Date Conversion to Character

**Usage**

```
## 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 <a href="#">RataDie</a> object.
...	Currently not used.
prefix	A <a href="#">character</a> 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 <a href="#">logical</a> scalar: should the label of the calendar be displayed?
calendar	A <a href="#">TimeScale</a> object specifying the target calendar (see <a href="#">calendar()</a> ).

**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()
```

---

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 <b>numeric</b> vector of (decimal) years or a <b>RataDie</b> object (see <b>fixed()</b> ) giving the beginning of the time intervals.
end	A <b>numeric</b> vector of (decimal) years or a <b>RataDie</b> object (see <b>fixed()</b> ) giving the end of the time intervals.
calendar	A <b>TimeScale</b> object specifying the calendar of time (see <b>calendar()</b> ). If missing, time must be a <b>RataDie</b> object.
...	Currently not used.
names	A <b>character</b> string specifying the names of the time series.
scale	A length-one <b>numeric</b> 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: [as.data.frame\(\)](#), [length\(\)](#), [subset\(\)](#)

---

length	<i>Length</i>
--------	---------------

---

**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: [as.data.frame\(\)](#), [labels\(\)](#), [subset\(\)](#)

---

overlap	<i>Time Overlap</i>
---------	---------------------

---

**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 <a href="#">TimeIntervals</a> object.
...	Currently not used.
calendar	A <a href="#">TimeScale</a> object specifying the target calendar (see <a href="#">calendar()</a> ). If NULL (the default), <i>rata die</i> are returned.
aggregate	A <a href="#">logical</a> 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

**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 <a href="#">TimeSeries</a> or a <a href="#">TimeIntervals</a> object.
calendar	A <a href="#">TimeScale</a> object specifying the target calendar (see <a href="#">calendar()</a> ).
groups	A <a href="#">character</a> vector specifying the group each interval belongs to.
sort	A <a href="#">logical</a> scalar: should the data be sorted in chronological order?

decreasing	A <b>logical</b> scalar: should the sort order be decreasing? Only used if sort is TRUE.
xlab, ylab	A <b>character</b> vector giving the x and y axis labels.
main	A <b>character</b> string giving a main title for the plot.
sub	A <b>character</b> string giving a subtitle for the plot.
ann	A <b>logical</b> scalar: should the default annotation (title and x and y axis labels) appear on the plot?
axes	A <b>logical</b> scalar: should axes be drawn on the plot?
frame.plot	A <b>logical</b> 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. <a href="#">graphical parameters</a> ).
facet	A <b>character</b> 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 <b>function</b> in the form function(x, y, ...) which gives the action to be carried out in each panel of the display. The default is <a href="#">graphics::lines()</a> .
flip	A <b>logical</b> 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 <b>integer</b> 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 <a href="#">RataDie</a> object.
calendar	A <a href="#">TimeScale</a> object specifying the target calendar (see <a href="#">calendar()</a> ).
...	Further parameters to be passed to <a href="#">base::pretty()</a> .

## 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	<i>RataDie</i>
---------------	----------------

---

**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](#)

---

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 <a href="#">numeric</a> vector, matrix or array of the observed time-series values. A <a href="#">data.frame</a> will be coerced to a numeric matrix via <a href="#">data.matrix()</a> .
time	A <a href="#">numeric</a> vector of (decimal) years or a <a href="#">RataDie</a> object (see <a href="#">fixed()</a> ).
calendar	A <a href="#">TimeScale</a> object specifying the calendar of time (see <a href="#">calendar()</a> ). If missing, time must be a <a href="#">RataDie</a> object.
...	Currently not used.
names	A <a href="#">character</a> string specifying the names of the time series.
scale	A length-one <a href="#">numeric</a> 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 <a href="#">TimeSeries</a> or a <a href="#">TimeIntervals</a> object.
...	Currently not used.
calendar	A <a href="#">TimeScale</a> object specifying the target calendar (see <a href="#">calendar()</a> ). If NULL (the default), <i>rata die</i> 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	<i>Terminal Times</i>
-------	-----------------------

---

### 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 <a href="#">TimeSeries</a> object.
calendar	A <a href="#">TimeScale</a> object specifying the target calendar (see <a href="#">calendar()</a> ). If NULL (the default), <i>rata die</i> 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

**See Also**

Other mutators: [as.data.frame\(\)](#), [labels\(\)](#), [length\(\)](#)

---

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**

<code>x</code>	A <a href="#">TimeSeries</a> object.
<code>calendar</code>	A <a href="#">TimeScale</a> object specifying the target calendar (see <a href="#">calendar()</a> ). If NULL (the default), <i>rata die</i> are returned.
<code>...</code>	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	<i>TimeScale</i>
-----------------	------------------

---

**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 <a href="#">TimeSeries</a> object.
start	A length-one <a href="#">numeric</a> vector specifying the start time of the period of interest.
end	A length-one <a href="#">numeric</a> 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 <a href="#">integer</a> 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 <a href="#">numeric</a> vector giving the points at which tick-marks are to be drawn. If NULL, tickmark locations are computed.
format	A <a href="#">character</a> 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 <a href="#">logical</a> scalar specifying whether annotations are to be made at the tick-marks, or a vector of <a href="#">character</a> strings to be placed at the tickpoints.
calendar	A <a href="#">TimeScale</a> object specifying the target calendar (see <a href="#">calendar()</a> ).
current_calendar	A <a href="#">TimeScale</a> object specifying the calendar used by the last call to <a href="#">plot()</a> .
...	Further parameters to be passed to <a href="#">graphics::axis()</a> . (e.g. <a href="#">graphical parameters</a> ).

**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, 41
- \* **calendar tools**
  - calendar, 10
  - calendar\_get, 12
  - convert, 14
  - get\_calendar, 21
  - gregorian, 22
  - is\_calendar, 25
  - julian, 26
- \* **chronological reasoning tools**
  - overlap, 28
- \* **classes**
  - RataDie-class, 33
  - TimeIntervals-class, 40
  - TimeScale-class, 41
  - TimeSeries-class, 42
- \* **datasets**
  - dates, 15
- \* **fixed date tools**
  - arithmetic, 3
  - as\_date, 5
  - as\_decimal, 7
  - as\_fixed, 8
  - as\_year, 9
  - fixed, 15
  - fixed\_gregorian, 17
  - fixed\_julian, 19
  - format, 20
  - pretty, 32
- \* **mutators**
  - as.data.frame, 4
  - labels, 27
  - length, 28
  - subset, 38
- \* **plotting tools**
  - image, 23
  - plot, 29
  - year\_axis, 43
- \* **time classes**
  - RataDie-class, 33
  - TimeIntervals-class, 40
  - TimeSeries-class, 42
- \* **time intervals**
  - intervals, 24
- \* **time series**
  - series, 34
- \* **tools**
  - span, 35
  - start, 37
  - time, 39
  - window, 42
  - .RataDie (RataDie-class), 33
  - .TimeIntervals (TimeIntervals-class), 40
  - .TimeScale (TimeScale-class), 41
  - .TimeSeries (TimeSeries-class), 42
  - [,RataDie-method (subset), 38
  - [,TimeIntervals-method (subset), 38
  - [,TimeSeries-method (subset), 38
  - AD (gregorian), 22
  - Arith,numeric,RataDie-method (arithmetic), 3
  - Arith,RataDie,numeric-method (arithmetic), 3
  - Arith,RataDie,RataDie-method (arithmetic), 3
  - arithmetic, 3, 6–8, 10, 16, 18, 19, 21, 33
  - array, 42
  - as.data.frame, 4, 27, 28, 39
  - as.data.frame,TimeIntervals-method (as.data.frame), 4
  - as.data.frame,TimeSeries-method (as.data.frame), 4
  - as\_date, 3, 5, 7, 8, 10, 16, 18, 19, 21, 33
  - as\_date,numeric,GregorianCalendar-method (as\_date), 5
  - as\_date,numeric,JulianCalendar-method (as\_date), 5

- as\_date-method (as\_date), 5
- as\_decimal, 3, 6, 7, 8, 10, 16, 18, 19, 21, 33
- 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, 3, 6, 7, 8, 10, 16, 18, 19, 21, 33
- as\_fixed, numeric-method (as\_fixed), 8
- as\_fixed-method (as\_fixed), 8
- as\_year, 3, 6–8, 9, 16, 18, 19, 21, 33
- as\_year, numeric, GregorianCalendar-method (as\_year), 9
- as\_year, numeric, JulianCalendar-method (as\_year), 9
- as\_year-method (as\_year), 9
  
- b2k (gregorian), 22
- base::pretty(), 32
- BC (gregorian), 22
- BCE (gregorian), 22
- BP (gregorian), 22
  
- calendar, 10, 13, 14, 22, 23, 26, 27
- calendar(), 4, 6, 7, 9, 16, 20, 22–25, 27, 29, 30, 32, 34, 36, 37, 39, 44
- 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, 22, 23, 26, 27
- 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), 22
- character, 10, 13, 20–22, 25, 27, 30, 31, 34, 40, 41, 44
- convert, 11, 13, 14, 22, 23, 26, 27
- convert, character, character-method (convert), 14
- convert, TimeScale, TimeScale-method (convert), 14
- convert-method (convert), 14
  
- data.frame, 4, 5, 15, 34
- data.frame(), 4
- data.matrix(), 34
- dates, 15
  
- end, TimeIntervals-method (start), 37
- end, TimeSeries-method (start), 37
- end-method (start), 37
- era::era(), 11
  
- fixed, 3, 6–8, 10, 15, 18, 19, 21, 33
- fixed(), 6, 9, 18, 19, 25, 34
- 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, [3](#), [6–8](#), [10](#), [16](#), [17](#), [19](#), [21](#), [33](#)
- fixed\_julian, [3](#), [6–8](#), [10](#), [16](#), [18](#), [19](#), [21](#), [33](#)
- 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](#)
- format, [3](#), [6–8](#), [10](#), [16](#), [18](#), [19](#), [20](#), [33](#)
- format, RataDie-method (format), [20](#)
- format, TimeScale-method (format), [20](#)
- frequency, TimeSeries-method (time), [39](#)
- frequency-method (time), [39](#)
- function, [14](#), [31](#)
  
- get\_calendar, [11](#), [13](#), [14](#), [21](#), [23](#), [26](#), [27](#)
- graphical parameters, [31](#), [44](#)
- graphics::axis(), [44](#)
- graphics::image(), [24](#)
- graphics::lines(), [31](#)
- graphics::plot(), [31](#)
- gregorian, [11](#), [13](#), [14](#), [22](#), [22](#), [26](#), [27](#)
- GregorianCalendar, [23](#)
  
- image, [23](#), [31](#), [44](#)
- image(), [22](#)
- image, TimeSeries-method (image), [23](#)
- integer, [13](#), [16](#), [24](#), [28](#), [31](#), [41](#), [44](#)
- intervals, [24](#)
- intervals, numeric, numeric, TimeScale-method (intervals), [24](#)
- intervals, RataDie, RataDie, missing-method (intervals), [24](#)
- intervals-method (intervals), [24](#)
- is\_calendar, [11](#), [13](#), [14](#), [22](#), [23](#), [25](#), [27](#)
- is\_gregorian (is\_calendar), [25](#)
- is\_julian (is\_calendar), [25](#)
  
- J (julian), [26](#)
- julian, [11](#), [13](#), [14](#), [22](#), [23](#), [26](#), [26](#)
- JulianCalendar, [26](#)
  
- labels, [5](#), [27](#), [28](#), [39](#)
- labels, TimeIntervals-method (labels), [27](#)
- labels, TimeSeries-method (labels), [27](#)
- length, [5](#), [27](#), [28](#), [39](#)
- length, TimeIntervals-method (length), [28](#)
  
- logical, [3](#), [9](#), [20](#), [26](#), [29–31](#), [38](#), [44](#)
- matrix, [29](#)
  
- numeric, [3](#), [6–9](#), [13](#), [16](#), [18](#), [19](#), [25](#), [33](#), [34](#), [36](#), [37](#), [39](#), [41](#), [43](#), [44](#)
  
- overlap, [28](#)
- overlap, TimeIntervals-method (overlap), [28](#)
- overlap-method (overlap), [28](#)
  
- plot, [24](#), [29](#), [44](#)
- plot(), [22](#), [44](#)
- plot, TimeIntervals, missing-method (plot), [29](#)
- plot, TimeSeries, missing-method (plot), [29](#)
- pretty, [3](#), [6–8](#), [10](#), [16](#), [18](#), [19](#), [21](#), [32](#)
- pretty, RataDie-method (pretty), [32](#)
  
- RataDie, [3](#), [6](#), [8](#), [9](#), [16](#), [18–20](#), [25](#), [32–34](#), [40](#), [42](#)
- RataDie-class, [33](#)
  
- series, [34](#)
- series, array, numeric, TimeScale-method (series), [34](#)
- series, array, RataDie, missing-method (series), [34](#)
- series, data.frame, numeric, TimeScale-method (series), [34](#)
- series, data.frame, RataDie, missing-method (series), [34](#)
- series, matrix, numeric, TimeScale-method (series), [34](#)
- series, matrix, RataDie, missing-method (series), [34](#)
- series, numeric, numeric, TimeScale-method (series), [34](#)
- series, numeric, RataDie, missing-method (series), [34](#)
- series-method (series), [34](#)
- set\_calendar (get\_calendar), [21](#)
- sign, [13](#)
- span, [35](#), [37](#), [39](#), [43](#)
- span, TimeIntervals-method (span), [35](#)
- span, TimeSeries-method (span), [35](#)
- span-method (span), [35](#)
- start, [36](#), [37](#), [39](#), [43](#)

start, TimeIntervals-method (start), 37  
start, TimeSeries-method (start), 37  
start-method (start), 37  
subset, 5, 27, 28, 38

time, 36, 37, 39, 43  
time, TimeSeries-method (time), 39  
time-method (time), 39  
TimeIntervals, 4, 25, 29, 30, 36  
TimeIntervals-class, 40  
TimeScale, 4, 6, 7, 9, 11, 13, 14, 16, 20, 22,  
24, 25, 29, 30, 32, 34, 36, 37, 39, 44  
TimeScale-class, 41  
TimeSeries, 4, 24, 30, 35–37, 39, 43  
TimeSeries-class, 42

window, 36, 37, 39, 42  
window, TimeSeries-method (window), 42  
window-method (window), 42

year\_axis, 24, 31, 43