Package 'nanotime'

December 15, 2021

Type Package

Title Nanosecond-Resolution Time Support for R

Version 0.3.5

Date 2021-12-14

Author Dirk Eddelbuettel and Leonardo Silvestri

Maintainer Dirk Eddelbuettel <edd@debian.org>

Description Full 64-bit resolution date and time functionality with nanosecond granularity is provided, with easy transition to and from the standard 'POSIXct' type. Three additional classes offer interval, period and duration functionality for nanosecond-resolution timestamps.

Imports methods, bit64, RcppCCTZ (>= 0.2.9), zoo

Suggests tinytest, data.table, xts

LinkingTo Rcpp, RcppCCTZ, RcppDate

License GPL (>= 2)

URL https://github.com/eddelbuettel/nanotime, https://dirk.eddelbuettel.com/code/nanotime.html

BugReports https://github.com/eddelbuettel/nanotime/issues

RoxygenNote 7.1.2

Collate 'nanotime.R' 'nanoival.R' 'nanoduration.R' 'nanoperiod.R' 'RcppExports.R'

Encoding UTF-8

NeedsCompilation yes

Repository CRAN

Date/Publication 2021-12-15 03:50:11 UTC

R topics documented:

`all.equal.nanoduration`

Test if Two Objects are (Nearly) Equal

Description

Compare `target` and `current` testing 'near equality'. If they are different, comparison is still made to some extent, and a report of the differences is returned. Do not use `all.equal` directly in `if` expressions—either use `isTRUE(all.equal(....))` or `identical` if appropriate.

Usage

```
## S3 method for class 'nanoduration'
all.equal(
  target,
  current,
  tolerance = sqrt(.Machine$double.eps),
  scale = NULL,
  countEQ = FALSE,
  formatFUN = function(err, what) format(err),
  ...,
  check.attributes = TRUE
```

```
)

## S4 method for signature 'nanoduration'
all.equal(
  target,
  current,
  tolerance = sqrt(.Machine$double.eps),
  scale = NULL,
  countEQ = FALSE,
  formatFUN = function(err, what) format(err),
  ...,
  check.attributes = TRUE
)
```

Arguments

target, current	nanoduration arguments to be compared
tolerance	numeric >= 0. Differences smaller than tolerance are not reported. The default value is close to 1.5e-8.
scale	NULL or numeric > 0, typically of length 1 or length(target). See 'Details'.
countEQ	logical indicating if the target == current cases should be counted when computing the mean (absolute or relative) differences. The default, FALSE may seem misleading in cases where target and current only differ in a few places; see the extensive example.
formatFUN	a function of two arguments, err, the relative, absolute or scaled error, and what, a character string indicating the _kind_ of error; maybe used, e.g., to format relative and absolute errors differently.
...	further arguments for different methods
check.attributes	logical indicating if the attributes of target and current (other than the names) should be compared.

See Also

identical, isTRUE, ==, and all for exact equality testing.

`all.equal.nanoival` *Test if Two Objects are (Nearly) Equal*

Description

Compare target and current testing 'near equality'. If they are different, comparison is still made to some extent, and a report of the differences is returned. Do not use all.equal directly in if expressions—either use isTRUE(all.equal(....)) or identical if appropriate.

Usage

```
## S3 method for class 'nanoival'
all.equal(target, current, ..., check.attributes = TRUE)

## S4 method for signature 'nanoival'
all.equal(target, current, ..., check.attributes = TRUE)
```

Arguments

target, current	nanoival arguments to be compared
...	further arguments for different methods
check.attributes	logical indicating if the attributes of target and current (other than the names) should be compared.

See Also

identical, isTRUE, ==, and all for exact equality testing.

---

all.equal.nanoperiod *Test if Two Objects are (Nearly) Equal*

Description

Compare target and current testing 'near equality'. If they are different, comparison is still made to some extent, and a report of the differences is returned. Do not use all.equal directly in if expressions—either use isTRUE(all.equal(....)) or identical if appropriate.

Usage

```
## S3 method for class 'nanoperiod'
all.equal(target, current, ..., check.attributes = TRUE)

## S4 method for signature 'nanoperiod'
all.equal(target, current, ..., check.attributes = TRUE)
```

Arguments

target, current	nanoperiod arguments to be compared
...	further arguments for different methods
check.attributes	logical indicating if the attributes of target and current (other than the names) should be compared.

See Also

identical, isTRUE, ==, and all for exact equality testing.

`all.equal.nanotime` *Test if Two Objects are (Nearly) Equal*

Description

Compare `target` and `current` testing 'near equality'. If they are different, comparison is still made to some extent, and a report of the differences is returned. Do not use `all.equal` directly in `if` expressions—either use `isTRUE(all.equal(....))` or `identical` if appropriate.

Usage

```
## S3 method for class 'nanotime'
all.equal(
  target,
  current,
  tolerance = sqrt(.Machine$double.eps),
  scale = NULL,
  countEQ = FALSE,
  formatFUN = function(err, what) format(err),
  ...,
  check.attributes = TRUE
)

## S4 method for signature 'nanotime'
all.equal(
  target,
  current,
  tolerance = sqrt(.Machine$double.eps),
  scale = NULL,
  countEQ = FALSE,
  formatFUN = function(err, what) format(err),
  ...,
  check.attributes = TRUE
)
```

Arguments

`target, current`	`nanotime` arguments to be compared
`tolerance`	numeric >= 0. Differences smaller than `tolerance` are not reported. The default value is close to `1.5e-8`.
`scale`	`NULL` or numeric > 0, typically of length 1 or `length(target)`. See 'Details'.

countEQ — logical indicating if the target == current cases should be counted when computing the mean (absolute or relative) differences. The default, FALSE may seem misleading in cases where target and current only differ in a few places; see the extensive example.

formatFUN — a function of two arguments, err, the relative, absolute or scaled error, and what, a character string indicating the _kind_ of error; maybe used, e.g., to format relative and absolute errors differently.

... — further arguments for different methods

check.attributes — logical indicating if the attributes of target and current (other than the names) should be compared.

See Also

identical, isTRUE, ==, and all for exact equality testing.

---

intersect,nanoival,nanoival-method
Set operations

Description

Performs set intersection, union and difference between vectors of temporal types from the nanotime package.

Usage

```
## S4 method for signature 'nanoival,nanoival'
intersect(x, y)

## S4 method for signature 'nanoival,nanoival'
union(x, y)

## S4 method for signature 'nanoival,nanoival'
setdiff(x, y)

## S4 method for signature 'nanotime,nanoival'
intersect.idx(x, y)

## S3 method for class 'nanotime'
x %in% table

## S4 method for signature 'nanotime,nanoival'
intersect(x, y)

## S4 method for signature 'nanotime,nanoival'
```

```
setdiff(x, y)

## S4 method for signature 'nanotime,nanoival'
setdiff.idx(x, y)

## S4 method for signature 'nanotime,nanotime'
intersect(x, y)

## S4 method for signature 'nanotime,nanotime'
union(x, y)

## S4 method for signature 'nanotime,nanotime'
setdiff(x, y)
```

Arguments

x, y	a temporal type
table	nanoival: used in %in%

Details

Set operations between nanoival operands allow the construction of complex interval vectors (i.e. a nanoival vector can specify any number of inclusions and exclusions of time). Set operations between nanotime and nanoival allow to subset time vectors with interval vectors. In addition to the generic set functions, the function intersect.idx is defined which returns the indices of the intersection, and the operator %in% is overloaded for nanotime-nanoival which returns a logical vector that indicates which elements belong to the interval vector.

Value

intersect, union, setdiff return temporal types that are the result of the intersection. For instance, set operations on two nanoival return a nanoival, whereas intersection between a nanoival and a nanotime returns a nanotime. intersect.idx return a list of vectors representing the element indices that intersect and setdiff.idx returns a vector representing the element indices to be removed.

Examples

```
## Not run:
## a vector of 'nanotime' can be subsetted by a 'nanoival' which is equivalent to 'intersect':
one_second <- 1e9
a <- seq(nanotime("2012-12-12 12:12:12+00:00"), length.out=10, by=one_second)
idx <- c(as.nanoival("-2012-12-12 12:12:10+00:00 -> 2012-12-12 12:12:14+00:00-"),
         as.nanoival("+2012-12-12 12:12:18+00:00 -> 2012-12-12 12:12:20+00:00+"))
a[idx]
intersect(a, idx)

## 'nanoival' also has the set operations 'union', 'intersect', 'setdiff':
a <- seq(nanotime("2012-12-12 12:12:12+00:00"), length.out=10, by=one_second)
i <- as.nanoival("-2012-12-12 12:12:14+00:00 -> 2012-12-12 12:12:18+00:00-")
```

```
setdiff(a, i)

i1 <- as.nanoival("+2012-12-12 12:12:14+00:00 -> 2012-12-12 12:12:17+00:00-")
i2 <- as.nanoival("+2012-12-12 12:12:16+00:00 -> 2012-12-12 12:12:18+00:00-")
union(i1, i2)

## 'intersect.idx' returns the indices of the intersection:
a <- seq(nanotime("2012-12-12 12:12:12+00:00"), length.out=10, by=one_second)
idx <- as.nanoival("+2012-12-12 12:12:14+00:00 -> 2012-12-12 12:12:19+00:00+")
idx_intersect <- intersect.idx(a, idx)

## Intersection can be performed using these indices:
a[idx_intersect$x]

## which is equivalent to:
a[idx]

## The logical vector indicating intersection can be obtained like this:
a %in% idx

## End(Not run)
```

`is.unsorted,nanoival-method`

Test if a `nanoival` *vector is Not Sorted*

Description

Test if an object is not sorted (in increasing order), without the cost of sorting it.

Usage

```
## S4 method for signature 'nanoival'
is.unsorted(x, na.rm = FALSE, strictly = FALSE)
```

Arguments

`x`	a `nanoival` vector
`na.rm`	logical. Should missing values be removed before checking?
`strictly`	logical indicating if the check should be for _strictly_ increasing values.

See Also

`sort`

nanoduration-class *Duration type with nanosecond precision*

Description

The type nanoduration is a length of time (implemented as an S4 class) with nanosecond precision. It is a count of nanoseconds and may be negative. The expected arithmetic operations are provided, including sequence generation.

Usage

```
nanoduration(hours, minutes, seconds, nanoseconds)

## S4 method for signature 'character'
as.nanoduration(x)

## S4 method for signature 'integer64'
as.nanoduration(x)

## S4 method for signature 'numeric'
as.nanoduration(x)

## S4 method for signature 'integer'
as.nanoduration(x)

## S4 method for signature '`NULL`'
as.nanoduration(x)

## S4 method for signature 'missing'
as.nanoduration(x)

## S4 method for signature 'nanoduration'
show(object)

## S4 method for signature 'nanoduration'
print(x, quote = FALSE, ...)

## S3 method for class 'nanoduration'
format(x, ...)

## S3 method for class 'nanoduration'
as.integer64(x, ...)

## S4 method for signature 'nanoduration'
as.character(x)

## S4 method for signature 'nanoduration'
```

```
is.na(x)

## S4 method for signature 'nanoduration,nanoduration'
e1 - e2

## S4 method for signature 'nanoduration,integer64'
e1 - e2

## S4 method for signature 'nanoduration,integer'
e1 - e2

## S4 method for signature 'nanoduration,numeric'
e1 - e2

## S4 method for signature 'nanoduration,ANY'
e1 - e2

## S4 method for signature 'nanotime,nanoduration'
e1 - e2

## S4 method for signature 'integer64,nanoduration'
e1 - e2

## S4 method for signature 'integer,nanoduration'
e1 - e2

## S4 method for signature 'numeric,nanoduration'
e1 - e2

## S4 method for signature 'ANY,nanoduration'
e1 - e2

## S4 method for signature 'nanoduration,ANY'
e1 + e2

## S4 method for signature 'nanoduration,nanoduration'
e1 + e2

## S4 method for signature 'nanoduration,integer64'
e1 + e2

## S4 method for signature 'nanoduration,numeric'
e1 + e2

## S4 method for signature 'nanotime,nanoduration'
e1 + e2

## S4 method for signature 'nanoival,nanoduration'
```

```
e1 + e2

## S4 method for signature 'integer64,nanoduration'
e1 + e2

## S4 method for signature 'numeric,nanoduration'
e1 + e2

## S4 method for signature 'nanoduration,numeric'
e1 * e2

## S4 method for signature 'nanoduration,integer64'
e1 * e2

## S4 method for signature 'numeric,nanoduration'
e1 * e2

## S4 method for signature 'integer64,nanoduration'
e1 * e2

## S4 method for signature 'nanoduration,nanoduration'
e1 / e2

## S4 method for signature 'nanoduration,integer64'
e1 / e2

## S4 method for signature 'nanoduration,numeric'
e1 / e2

## S4 method for signature 'nanoduration,ANY'
Arith(e1, e2)

## S4 method for signature 'nanoduration,character'
Compare(e1, e2)

## S4 method for signature 'character,nanoduration'
Compare(e1, e2)

## S4 method for signature 'nanoduration,ANY'
Compare(e1, e2)

## S4 method for signature 'nanoduration'
abs(x)

## S4 method for signature 'nanoduration'
sign(x)

## S4 method for signature 'nanoduration'
```

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

## S4 method for signature 'nanoduration'
min(x, ..., na.rm = FALSE)

## S4 method for signature 'nanoduration'
max(x, ..., na.rm = FALSE)

## S4 method for signature 'nanoduration'
range(x, ..., na.rm = FALSE)

## S4 method for signature 'nanoduration'
x[[i, j, ..., drop = FALSE]]

## S4 method for signature 'nanoduration,numeric'
x[i, j, ..., drop = FALSE]

## S4 method for signature 'nanoduration,logical'
x[i, j, ..., drop = FALSE]

## S4 method for signature 'nanoduration,character'
x[i, j, ..., drop = FALSE]

## S4 method for signature 'nanoduration,ANY'
x[i, j, ..., drop = FALSE]

## S4 replacement method for signature 'nanoduration,ANY,ANY,ANY'
x[i, j, ...] <- value

## S3 method for class 'nanoduration'
c(...)

NA_nanoduration_
```

Arguments

hours	number of hours
minutes	number of minutes
seconds	number of seconds
nanoseconds	number of nanoseconds
x	a nanoduration object
object	argument for method show
quote	indicates if the output of print should be quoted
...	further arguments passed to or from methods.
e1	Operand of class nanoival
e2	Operand of class nanoival

na.rm	if TRUE NA values are removed for the computation
i	index specifying elements to extract or replace.
j	Required for [signature but ignored here
drop	Required for [signature but ignored here
value	argument for nanoduration-class

Format

An object of class nanoduration of length 1.

Details

A nanoduration can be constructed with the function as.nanoduration which can take the types integer64, integer and numeric (all indicating the count in nanosecond units) or the type character.

It can also be constructed by specifying with individual arguments the hours, minutes, seconds and nanoseconds with a call to nanoduration.

A nanoduration is displayed as hours, minutes, seconds and nanoseconds like this: 110:12:34.123_453_001. The nanosecond precision displayed is adjusted as necessary, so e.g. 1 second is displayed as 00:00:01.

Value

A nanoduration object

Author(s)

Dirk Eddelbuettel

Leonardo Silvestri

See Also

nanotime

Examples

```
## constructors:
nanoduration(hours=10, minutes=3, seconds=2, nanoseconds=999999999)
as.nanoduration("10:03:02.999_999_999")
as.nanoduration(36182999999999)

## arithmetic:
as.nanoduration(10e9) - as.nanoduration(9e9)
as.nanoduration(10e9) + as.nanoduration(-9e9)
as.nanoduration("24:00:00") / 2
as.nanoduration("24:00:00") / as.nanoduration("12:00:00")

## comparison:
as.nanoduration("10:03:02.999_999_999") == 36182999999999
as.nanoduration("10:03:02.999_999_999") > as.nanoduration("10:03:02.999_999_998")
```

```
as.nanoduration("10:03:02.999_999_998") < "10:03:02.999_999_999"
```

---

nanoival-class *Interval type with nanosecond precision*

Description

nanoival is a time interval type (an S4 class) with nanosecond precision. One of its purposes is to allow quick subsetting of a nanotime vector. nanoival is composed of a nanotime pair which defines the start and end of the time interval. Additionally, it has a pair of logical values which determine if the start and end of the time interval are open (true) or closed (false).

Usage

```
nanoival(start, end, sopen = FALSE, eopen = TRUE)

## S4 method for signature 'nanoival'
nanoival.start(x)

## S4 method for signature 'nanoival'
nanoival.end(x)

## S4 method for signature 'nanoival'
nanoival.sopen(x)

## S4 method for signature 'nanoival'
nanoival.eopen(x)

## S3 method for class 'nanoival'
format(x, ...)

## S4 method for signature 'nanoival'
print(x, quote = FALSE, ...)

## S4 method for signature 'nanoival'
show(object)

## S4 method for signature 'character'
as.nanoival(from, format = "", tz = "")

## S4 method for signature '`NULL`'
as.nanoival(from, format = "", tz = "")

## S4 method for signature 'missing'
as.nanoival(from, format = "", tz = "")
```

```
## S4 method for signature 'nanoival'
is.na(x)

## S4 replacement method for signature 'nanoival'
is.na(x) <- value

## S4 method for signature 'nanoival,nanoival'
e1 < e2

## S4 method for signature 'nanoival,nanoival'
e1 <= e2

## S4 method for signature 'nanoival,nanoival'
e1 > e2

## S4 method for signature 'nanoival,nanoival'
e1 >= e2

## S4 method for signature 'nanoival,nanoival'
e1 == e2

## S4 method for signature 'nanoival,nanoival'
e1 != e2

## S4 method for signature 'nanoival,integer64'
e1 - e2

## S4 method for signature 'nanoival,numeric'
e1 - e2

## S4 method for signature 'nanoival,integer64'
e1 + e2

## S4 method for signature 'nanoival,numeric'
e1 + e2

## S4 method for signature 'integer64,nanoival'
e1 + e2

## S4 method for signature 'numeric,nanoival'
e1 + e2

## S4 method for signature 'nanoival'
x[[i, j, ..., drop = FALSE]]

## S4 method for signature 'nanoival,logical'
x[i, j, ..., drop = FALSE]
```

```
## S4 method for signature 'nanoival,numeric'
x[i, j, ..., drop = FALSE]

## S4 method for signature 'nanoival,character'
x[i, j, ..., drop = FALSE]

## S4 method for signature 'nanoival,ANY'
x[i, j, ..., drop = FALSE]

## S4 replacement method for signature 'nanoival,logical,ANY,nanoival'
x[i, j, ...] <- value

## S3 method for class 'nanoival'
c(...)

## S4 method for signature 'nanoival'
t(x)

## S4 method for signature 'nanotime,nanoival'
x[i, j, ..., drop = TRUE]

NA_nanoival_

## S3 method for class 'nanoival'
as.character(x, ...)
```

Arguments

start	nanotime start of interval
end	nanotime end of interval
sopen	logical indicating if the start of the interval is open
eopen	logical indicating if the end of the interval is open
x, from	a nanoival object
...	further arguments passed to or from methods.
quote	indicates if the output of print should be quoted
object	argument for method show
format	A character string. Can also be set via options("nanotimeFormat") and uses '%Y-%m-%dT%H:%M:%E9S%Ez' as a default and fallback
tz	character indicating a timezone
value	argument for nanoival-class
e1	Operand of class nanoival
e2	Operand of class nanoival
i	index specifying elements to extract or replace.
j	Required for [signature but ignored here
drop	Required for [signature but ignored here

Format

An object of class nanoival of length 1.

Details

An interval object can be constructed with the constructor nanoival which takes as arguments two nanotime objects that define the start and the end of the interval, together with two logical arguments that define if the start and the end of the interval are open (true) or closed (false) (note that these objects can all be vector, and therefore the interval object is not necessarily scalar). Alternatively, an interval can be constructed with a character: the format follows that of nanotime; the start time is preceeded by either - or + indicating if the interval start is open (-) or closed (+); the start and end times are separated by an arrow ->; the end is folloed by either - or + which have the same semantics as the start time.

The most important set of methods defined for interval are set functions intersect, union and setdiff.

Additionally, interval allows the subsetting into a nanotime vector. Note that subsetting is allowed only if the nanotime vector is sorted.

Finally, accessors are provided to get the interval start (start), the end (end), the open/close status of the start (sopen) and the open/close status of the end (eopen). The former return a nanotime while the latter return a logical.

Value

A nanoival object

Output Format

Formatting and character conversion for nanoival objects is identical to nanotime objects. The default format is ISO3339 compliant: %Y-%m-%dT%H:%M:%E9S%Ez. It specifies a standard ISO 8601 part for date and time — as well as nine digits of precision for fractional seconds (down to nanoseconds) and on offset (typically zero as we default to UTC). It can be overriden by using options() with the key of nanotimeFormat and a suitable value. Similarly, nanotimeTz can be used to select a different timezone.

Author(s)

Dirk Eddelbuettel

Leonardo Silvestri

See Also

intersect.idx, setdiff.idx,

Examples

```
## Not run:
## creating a \code{nanoival}, with the start time included ('+') and the end
## time excluded ('-')
```

```
as.nanoival("+2012-03-01T21:21:00.000000001+00:00->2015-01-01T21:22:00.000000999+04:00-")

## a \code{nanoival} can also be created with a pair of \code{nanotime} objects, a start
## and an end, and optionally two logicals determining if the interval start(end) are open
## or closed; by default the start is closed and end is open:
start <- nanotime("2012-03-01T21:21:00.000000001+00:00")
end <- nanotime("2013-03-01T21:21:00.000000001+00:00")
nanoival(start, end)

## a vector of 'nanotime' can be subsetted by a 'nanoival':
one_second <- 1e9
a <- seq(nanotime("2012-12-12 12:12:12+00:00"), length.out=10, by=one_second)
idx <- c(as.nanoival("-2012-12-12 12:12:10+00:00 -> 2012-12-12 12:12:14+00:00-"),
         as.nanoival("+2012-12-12 12:12:18+00:00 -> 2012-12-12 12:12:20+00:00+"))
a[idx]

## End(Not run)
```

---

nanoperiod-class *Period type with nanosecond precision*

Description

nanoperiod is a length of time type (implemented as an S4 class) with nanosecond precision. It differs from nanoduration because it is capable of representing calendar months and days. It can thus represent years (12 months) and weeks (7 days). A period is a somewhat abstract representation of time: it is only when anchored to a point in time and in a specific time zone that it is possible to convert it to a specific duration. This means that many of the operations involving periods need the additional argument tz.

Usage

```
nanoperiod(months = 0, days = 0, duration = as.nanoduration(0))

## S4 method for signature 'character'
as.nanoperiod(x)

## S4 method for signature 'integer64'
as.nanoperiod(x)

## S4 method for signature 'numeric'
as.nanoperiod(x)

## S4 method for signature 'integer'
as.nanoperiod(x)

## S4 method for signature 'nanoduration'
as.nanoperiod(x)
```

```
## S4 method for signature '`NULL`'
as.nanoperiod(x)

## S4 method for signature 'missing'
as.nanoperiod(x)

## S4 method for signature 'nanoperiod'
show(object)

## S4 method for signature 'nanoperiod'
print(x, quote = FALSE, ...)

## S3 method for class 'nanoperiod'
format(x, ...)

## S4 method for signature 'nanoperiod'
as.character(x)

## S4 method for signature 'nanoperiod'
is.na(x)

## S4 replacement method for signature 'nanoperiod'
is.na(x) <- value

## S4 method for signature 'nanoperiod'
x[[i, j, ..., drop = FALSE]]

## S4 method for signature 'nanoperiod,numeric'
x[i, j, ..., drop = FALSE]

## S4 method for signature 'nanoperiod,logical'
x[i, j, ..., drop = FALSE]

## S4 method for signature 'nanoperiod,character'
x[i, j, ..., drop = FALSE]

## S4 method for signature 'nanoperiod,ANY'
x[i, j, ..., drop = FALSE]

## S4 replacement method for signature 'nanoperiod,ANY,ANY,ANY'
x[i, j, ...] <- value

## S3 method for class 'nanoperiod'
c(...)

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

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

## S4 method for signature 'nanoperiod,ANY'
e1 - e2

## S4 method for signature 'nanoperiod,nanoperiod'
e1 - e2

## S4 method for signature 'nanoperiod,nanoduration'
e1 - e2

## S4 method for signature 'nanoperiod,integer64'
e1 - e2

## S4 method for signature 'nanoperiod,numeric'
e1 - e2

## S4 method for signature 'nanoduration,nanoperiod'
e1 - e2

## S4 method for signature 'integer64,nanoperiod'
e1 - e2

## S4 method for signature 'numeric,nanoperiod'
e1 - e2

## S4 method for signature 'nanoperiod,ANY'
e1 + e2

## S4 method for signature 'nanoperiod,nanoperiod'
e1 + e2

## S4 method for signature 'nanoperiod,nanoduration'
e1 + e2

## S4 method for signature 'nanoperiod,integer64'
e1 + e2

## S4 method for signature 'nanoperiod,nanotime'
e1 + e2

## S4 method for signature 'nanoival,nanoperiod'
e1 + e2

## S4 method for signature 'nanoperiod,nanoival'
e1 + e2
```

```
## S4 method for signature 'nanotime,nanoperiod'
e1 + e2

## S4 method for signature 'nanoperiod,numeric'
e1 + e2

## S4 method for signature 'nanoduration,nanoperiod'
e1 + e2

## S4 method for signature 'integer64,nanoperiod'
e1 + e2

## S4 method for signature 'numeric,nanoperiod'
e1 + e2

## S4 method for signature 'nanoperiod,integer64'
e1 * e2

## S4 method for signature 'nanoperiod,numeric'
e1 * e2

## S4 method for signature 'integer64,nanoperiod'
e1 * e2

## S4 method for signature 'numeric,nanoperiod'
e1 * e2

## S4 method for signature 'nanoperiod,integer64'
e1 / e2

## S4 method for signature 'nanoperiod,numeric'
e1 / e2

## S4 method for signature 'nanoperiod,nanoperiod'
e1 == e2

## S4 method for signature 'nanoperiod,nanoperiod'
e1 != e2

## S4 method for signature 'nanotime,nanoperiod,character'
plus(e1, e2, tz)

## S4 method for signature 'nanoperiod,nanotime,character'
plus(e1, e2, tz)

## S4 method for signature 'nanotime,nanoperiod,character'
minus(e1, e2, tz)
```

```
## S4 method for signature 'nanoperiod,nanotime,character'
minus(e1, e2, tz)

## S4 method for signature 'nanoival,nanoperiod,character'
plus(e1, e2, tz)

## S4 method for signature 'nanoperiod,nanoival,character'
plus(e1, e2, tz)

## S4 method for signature 'nanoival,nanoperiod,character'
minus(e1, e2, tz)

NA_nanoperiod_
```

Arguments

months	Used in the constructor to indicate the number of months of the nanoperiod
days	Used in the constructor to indicate the number of days of the nanoperiod
duration	Used in the constructor to indicate the duration component of the nanoperiod
x, value	An object of class nanoperiod
object	argument for method show
quote	indicates if the output of print should be quoted
...	further arguments
i	index specifying elements to extract or replace.
j	Required for [signature but ignored here
drop	Required for [signature but ignored here
e1	Operand of class nanoperiod
e2	Operand of class nanoperiod
tz	character indicating a timezone

Format

An object of class nanoperiod of length 1.

Constructors

The true constructor is

Output Format

A nanoperiod is displayed as months, days, and nanoduration like this: 10m2d/10:12:34.123_453_000.

Details

Adding or subtracting nanoperiod and nanotime require a timezone as third argument. For this reason it is not possible to use the binary operator 'code+'. Instead the functions 'plus' and 'minus' are defined. These functions attempt to keep the same offset within a day in the specified timezone: this means for instance that adding a day when that day crosses a time zone adjustment such as a daylight saving time, results in a true time increment of less or more than 24 hours to preserve the offset. Preserving the offset works for increments that are smaller than a day too, provided the increment results in a datetime where the timezone adjustment is valid. When this is not the case, adding a 'nanoperiod' behaves in the same way as adding a 'nanoduration'.

Author(s)

Dirk Eddelbuettel

Leonardo Silvestri

See Also

nanotime, nanoduration, nanoival, nanoperiod.month,nanoperiod-method

Examples

```
## Not run:
p <- nanoperiod(months=12, days=7, duration="01:00:00")
print(p)

#  when adding a \code{nanoperiod} to a \code{nanotime} or to a
# \code{nanoival}, a time zone must be specified:
y <- nanotime("1970-01-01T00:00:00+00:00")
plus(y, p, tz="America/Chicago")

## End(Not run)
```

---

nanoperiod.month,nanoperiod-method

Nanoperiod accessors

Description

These functions allow access to the components of a nanoperiod

Usage

```
## S4 method for signature 'nanoperiod'
nanoperiod.month(x)

## S4 method for signature 'nanoperiod'
nanoperiod.day(x)
```

```
## S4 method for signature 'nanoperiod'
nanoperiod.nanoduration(x)
```

Arguments

x	A nanoperiod

Value

nanoperiod.month and nanoperiod.day return an integer64 whereas nanoperiod.nanoduration returns a nanoduration

Author(s)

Dirk Eddelbuettel

Leonardo Silvestri

See Also

nanoduration

Examples

```
p <- as.nanoperiod("2y1m1d/12:00:00")
nanoperiod.month(p)
nanoperiod.day(p)
nanoperiod.nanoduration(p)
```

---

nanotime-class *Nanosecond resolution datetime functionality*

Description

Functions to operate on nanosecond time resolution using integer64 bit representation. Conversion functions for several standard R types are provided, and more will be added as needed.

Usage

```
nanotime(from, ...)

as.nanotime(from, ...)

## S4 method for signature 'character'
nanotime(from, format = "", tz = "")

## S4 method for signature 'character'
as.nanotime(from, format = "", tz = "")
```

```
nanotime.matrix(x)

## S4 method for signature 'POSIXct'
nanotime(from)

## S4 method for signature 'POSIXct'
as.nanotime(from)

## S4 method for signature 'POSIXlt'
nanotime(from)

## S4 method for signature 'POSIXlt'
as.nanotime(from)

## S4 method for signature 'Date'
nanotime(from)

## S4 method for signature 'Date'
as.nanotime(from)

## S4 method for signature 'nanotime'
print(x, format = "", tz = "", quote = FALSE, ...)

## S4 method for signature 'nanotime'
show(object)

## S3 method for class 'nanotime'
format(x, format = "", tz = "", ...)

## S3 method for class 'nanotime'
index2char(x, ...)

## S3 method for class 'nanotime'
as.POSIXct(x, tz = "", ...)

## S3 method for class 'nanotime'
as.POSIXlt(x, tz = "", ...)

## S3 method for class 'nanotime'
as.Date(x, ...)

## S3 method for class 'nanotime'
as.data.frame(x, ...)

## S3 method for class 'nanotime'
as.integer64(x, ...)
```

```
## S4 method for signature 'nanotime,character'
e1 - e2

## S4 method for signature 'nanotime,nanotime'
e1 - e2

## S4 method for signature 'nanotime,integer64'
e1 - e2

## S4 method for signature 'nanotime,numeric'
e1 - e2

## S4 method for signature 'ANY,nanotime'
e1 - e2

## S4 method for signature 'nanotime,ANY'
e1 - e2

## S4 method for signature 'nanotime,ANY'
e1 + e2

## S4 method for signature 'nanotime,integer64'
e1 + e2

## S4 method for signature 'nanotime,numeric'
e1 + e2

## S4 method for signature 'ANY,nanotime'
e1 + e2

## S4 method for signature 'integer64,nanotime'
e1 + e2

## S4 method for signature 'numeric,nanotime'
e1 + e2

## S4 method for signature 'nanotime,nanotime'
e1 + e2

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

## S4 method for signature 'nanotime,ANY'
Arith(e1, e2)

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

```
## S4 method for signature 'nanotime,character'
Compare(e1, e2)

## S4 method for signature 'character,nanotime'
Compare(e1, e2)

## S4 method for signature 'nanotime,POSIXt'
Compare(e1, e2)

## S4 method for signature 'POSIXt,nanotime'
Compare(e1, e2)

## S4 method for signature 'nanotime,ANY'
Compare(e1, e2)

## S4 method for signature 'nanotime,ANY'
Logic(e1, e2)

## S4 method for signature 'ANY,nanotime'
Logic(e1, e2)

## S4 method for signature 'nanotime'
Math(x)

## S4 method for signature 'nanotime'
Math2(x, digits)

## S4 method for signature 'nanotime'
Summary(x, ..., na.rm = FALSE)

## S4 method for signature 'nanotime'
min(x, ..., na.rm = FALSE)

## S4 method for signature 'nanotime'
max(x, ..., na.rm = FALSE)

## S4 method for signature 'nanotime'
range(x, ..., na.rm = FALSE)

## S4 method for signature 'nanotime'
Complex(z)

## S4 method for signature 'nanotime'
x[[i, j, ..., drop = FALSE]]

## S4 method for signature 'nanotime,numeric'
x[i, j, ..., drop = FALSE]
```

```
## S4 method for signature 'nanotime,logical'
x[i, j, ..., drop = FALSE]

## S4 method for signature 'nanotime,character'
x[i, j, ..., drop = FALSE]

## S4 method for signature 'nanotime,ANY'
x[i, j, ..., drop = FALSE]

## S4 replacement method for signature 'nanotime,ANY,ANY,ANY'
x[i, j, ...] <- value

## S3 method for class 'nanotime'
c(...)

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

## S4 method for signature 'nanotime'
is.na(x)

NA_nanotime_

## S3 method for class 'nanotime'
as.character(x, ...)

## S3 method for class 'nanoduration'
as.data.frame(x, ...)
```

Arguments

...	further arguments passed to or from methods.
format	A character string. Can also be set via options("nanotimeFormat") and uses '%Y-%m-%dT%H:%M:%E9S%Ez' as a default and fallback
tz	character specifying a timezone which is required for as.POSIXct, as.POSIXlt and can be specified for as.nanotime, format and print; it can also be set via options("nanotimeTz") and uses 'UTC' as a default and fallback
x, from	nanotime objects
quote	indicates if the output of print should be quoted
object	argument for method show
e1	Operand of class nanotime
e2	Operand of class nanotime
digits	Required for Math2 signature but ignored here
na.rm	a logical indicating whether missing values should be removed.
z	Required for Complex signature but ignored here
i	index specifying elements to extract or replace.

j	Required for [signature but ignored here
drop	Required for [signature but ignored here
value	argument for nanotime-class

Format

An object of class nanotime of length 1.

Details

Notice that the conversion from POSIXct explicitly sets the last three digits to zero. Nanosecond time stored in a 64-bit integer has nineteen digits precision where doubles (which are used internally for POSIXct as well) only have sixteen digits. So rather than showing three more (essentially *random*) digits it is constructed such that these three additional digits are zeros.

Value

A nanotime object

Caveats

Working with dates and times is *difficult*. One needs a representation of both *time points* and *time duration*. In R, think of Date or POSIXct objects for the former, and difftime for the later. Here we have time points nanotime, an interval type nanoival and two flavors of duration which are a simple count of nanoseconds nanoduration and a calendar duration that is able to track concepts such as months and days nanoperiod. Point in time and intervals are all based on durations relative to the epoch of January 1, 1970.

Input and Output Format

Formatting and character conversion for nanotime objects is done by functions from the RcppCCTZ package relying on code from its embedded CCTZ library. The default format is ISO3339 compliant: %Y-%m-%dT%H:%M:%E9S%Ez. It specifies a standard ISO 8601 part for date and time — as well as nine digits of precision for fractional seconds (down to nanoseconds) and on offset (typically zero as we default to UTC). It can be overriden by using options() with the key of nanotimeFormat and a suitable value. Similarly, nanotimeTz can be used to select a different timezone.

For input, some slack it cut, and various shortened formats are accepted by default such as 2020-03-10 or 2020-03-10 18:16:00, or 2020-03-10 18:16:00.001 (and the 'T' separator is optional.

tz parameter usage in constructors

The tz parameter is allowed only when constructing a nanotime from a character. This is because any numeric, Date and POSIXct is de facto considered an offset since the epoch. On the contrary, a character is considered interpretable and hence if it does not contain a timezone in its representation, it is possible to specify the tz argument to specify in which timezone it should be interpreted. This is useful in particular if one wants to convert a Date to be aligned to the beginning of the day in a specific timezone; in this case one should convert the Date to a character before calling the nanotime constructor with the desired timezone.

Author(s)

Dirk Eddelbuettel

Leonardo Silvestri

See Also

nanoival, nanoduration, nanoperiod, seq.nanotime

Examples

```
## Not run:
x <- nanotime(1)
print(x)
as.nanotime("1970-01-01T00:00:00.000000001+00:00")
as.nanotime("2020-03-10 Europe/Berlin")
as.nanotime("2020-03-10 18:31:23.001", tz="America/New_York")
x <- x + 1
print(x)
format(x)
x <- x + 10
print(x)
format(x)
nanotime(Sys.time()) + 1:3  # three elements each 1 ns apart
seq(x, by=as.nanoperiod("1d"), length.out=5, tz="Asia/Tokyo")

## End(Not run)
```

---

nano_ceiling *Rounding down or up a* nanotime *type*

Description

The functions nano_floor and nano_ceiling round down or up, respectively. Although the underlying implementation of nanotime has negative numbers for values before 1970-01-01 UTC, the rounding is always done backward in time for nano_floor and forward in time for nano_ceiling. The functions take a nanotime argument x which is the instance to round, together with a second argument precision which indicates an arbitrary precision to which the rounding should be performed. This argument can be either a nanoduration or or a nanoperiod. In the latter case, the argument tz must also be specified in order to give the nanoperiod a meaning. Finally, the nanotime argument origin can be optionally specified to fix the rounding to a specific point in time.

Usage

```
nano_ceiling(x, precision, ...)

nano_floor(x, precision, ...)
```

```
## S4 method for signature 'nanotime,nanoduration'
nano_ceiling(x, precision, origin = nanotime())

## S4 method for signature 'nanotime,nanoduration'
nano_floor(x, precision, origin = nanotime())

## S4 method for signature 'nanotime,nanoperiod'
nano_ceiling(x, precision, origin = nanotime(), tz)

## S4 method for signature 'nanotime,nanoperiod'
nano_floor(x, precision, origin = nanotime(), tz)
```

Arguments

x	a nanotime object which must be sorted
precision	a nanoduration or nanoperiod object indicating the rounding precision
...	for future additional arguments
origin	a nanotime scalar indicating the origin at which the rounding is considered
tz	a character scalar indicating the time zone in which to conduct the rounding

Details

This flexible rounding must be understood in the context of a vector. The rounding precision can then be considered as an interval that defines a grid over which the elements are either assigned to the starting value of the interval to which they belong (nano_floor) or the ending value of the interval to which they belong (nano_ceiling). This allows for a grouping of a nanotime vector on which a statistic may then be run. In the examples below, such a use case is shown in the context of a data.table object.

If "business" concepts such as month or days are needed, the argument precision must be of type period. It is then mandatory to specify the timezone argument tz as this ensures timezone correctness of the intervals including for example for the rare hourly transitions of some countries going from a timezone with a whole hour difference with UTC to one with a fractional hour difference. In the case of a period, the functions align the rounding if the precision is an integer divisor of a larger quantity. For instance, if one specifies a rounding of 6 hours, a divisor of a day, the hours are aligned on days and the rounding is made to a grid at hours 0, 6, 12 and 18 in the specified timezone. If the precision is not a divisor, the grid is aligned to the nearest hour before the first element of the vector to round.

The argument origin controls the reference point of the rounding, allowing arbitrary specification of the reference point of the rounding.

Examples

```
## Not run:
## "classic" rounding:
nano_floor(as.nanotime("2010-10-10 11:12:15 UTC"), as.nanoduration("01:00:00"))
## rounding with arbitrary precision:
nano_floor(as.nanotime("2010-10-10 11:12:15 UTC"), as.nanoduration("06:00:00"))
nano_floor(as.nanotime("2010-10-10 11:23:15 UTC"), as.nanoduration("00:15:00"))
```

```
nano_ceiling(as.nanotime("2010-10-10 11:23:15 UTC"), as.nanoduration("01:15:23"))
## controlling the reference point via the 'origin' argument:
nano_ceiling(as.nanotime("2010-10-10 11:23:15 UTC"),
             as.nanoduration("01:15:23"),
             origin=as.nanotime("2010-10-10 11:23:15 UTC"))
## using business concepts and rounding across a daylight saving change:
v <- seq(as.nanotime("2020-03-08 America/New_York"),
         by=as.nanoperiod("06:00:00"), length.out=8, tz="America/New_York")
print(nano_floor(v, as.nanoperiod("1d"), tz="America/New_York"), tz="America/New_York")
## using the concept in a 'data.table':
library(data.table)
n <- 3 * 24
idx <- seq(as.nanotime("2020-03-07 America/New_York"),
           by=as.nanoperiod("01:00:00"), length.out=n, tz="America/New_York")
dt <- data.table(idx, a=1:n, b=2:(n+1))
dt_mean <- dt[, list(mean = mean(a)),
              by=nano_ceiling(idx, as.nanoperiod("1d"), tz="America/New_York")]

## End(Not run)
```

---

nano_wday *Get a component of a date time*

Description

Get a component of a date time. nano_wday returns the numeric position in a week, with Sunday == 0. nano_mday returns the numeric day (i.e. a value from 1 to 31). nano_month returns the month (i.e. a value from 1 to 12). nano_year returns the year.

Usage

```
nano_wday(x, tz)

nano_mday(x, tz)

nano_month(x, tz)

nano_year(x, tz)
```

Arguments

x	a nanotime object
tz	character a string representing a timezone

Details

Note that the tz parameter is mandatory because the day boundary is different depending on the time zone and nanotime does not store the timezone as it is just an offset in nanoseconds from the epoch.

Examples

```
## Not run:
nano_wday(as.nanotime("2020-03-14 23:32:00-04:00"), "America/New_York")
nano_wday(as.nanotime("2020-03-14 23:32:00 America/New_York"), "Europe/Paris")
nano_mday(as.nanotime("2020-03-14 23:32:00-04:00"), "America/New_York")
nano_mday(as.nanotime("2020-03-14 23:32:00 America/New_York"), "Europe/Paris")
nano_month(as.nanotime("2020-12-31 23:32:00-04:00"), "America/New_York")
nano_month(as.nanotime("2020-12-31 23:32:00 America/New_York"), "Europe/Paris")
nano_year(as.nanotime("2020-12-31 23:32:00-04:00"), "America/New_York")
nano_year(as.nanotime("2020-12-31 23:32:00 America/New_York"), "Europe/Paris")

## End(Not run)
```

---

rep,nanoduration-method

Replicate Elements

Description

Replicates the values in 'x' similarly to the default method.

Usage

```
## S4 method for signature 'nanoduration'
rep(x, ...)
```

Arguments

x	a vector of nanoduration
...	further arguments: 'times' an integer-valued vector giving the (non-negative) number of times to repeat each element if of length 'length(x)', or to repeat the whole vector if of length 1. Negative or 'NA' values are an error. A 'double' vector is accepted, other inputs being coerced to an integer or double vector. 'length.out' non-negative integer. The desired length of the output vector. Other inputs will be coerced to a double vector and the first element taken. Ignored if 'NA' or invalid. 'each' non-negative integer. Each element of 'x' is repeated 'each' times. Other inputs will be coerced to an integer or double vector and the first element taken. Treated as '1' if 'NA' or invalid.

rep,nanoival-method *Replicate Elements*

Description

Replicates the values in 'x' similarly to the default method.

Usage

```
## S4 method for signature 'nanoival'
rep(x, ...)
```

Arguments

x	a vector of nanoival
...	further arguments: 'times' an integer-valued vector giving the (non-negative) number of times to repeat each element if of length 'length(x)', or to repeat the whole vector if of length 1. Negative or 'NA' values are an error. A 'double' vector is accepted, other inputs being coerced to an integer or double vector. 'length.out' non-negative integer. The desired length of the output vector. Other inputs will be coerced to a double vector and the first element taken. Ignored if 'NA' or invalid. 'each' non-negative integer. Each element of 'x' is repeated 'each' times. Other inputs will be coerced to an integer or double vector and the first element taken. Treated as '1' if 'NA' or invalid.

rep,nanoperiod-method *Replicate Elements*

Description

Replicates the values in 'x' similarly to the default method.

Usage

```
## S4 method for signature 'nanoperiod'
rep(x, ...)
```

Arguments

x	a vector of nanoperiod
...	further arguments: 'times' an integer-valued vector giving the (non-negative) number of times to repeat each element if of length 'length(x)', or to repeat the whole vector if of length 1. Negative or 'NA' values are an error. A 'double' vector is accepted, other inputs being coerced to an integer or double vector. 'length.out' non-negative integer. The desired length of the output vector. Other inputs will be coerced to a double vector and the first element taken. Ignored if 'NA' or invalid. 'each' non-negative integer. Each element of 'x' is repeated 'each' times. Other inputs will be coerced to an integer or double vector and the first element taken. Treated as '1' if 'NA' or invalid.

---

rep,nanotime-method *Replicate Elements*

Description

Replicates the values in 'x' similarly to the default method.

Usage

```
## S4 method for signature 'nanotime'
rep(x, ...)
```

Arguments

x	a vector of nanotime
...	further arguments: 'times' an integer-valued vector giving the (non-negative) number of times to repeat each element if of length 'length(x)', or to repeat the whole vector if of length 1. Negative or 'NA' values are an error. A 'double' vector is accepted, other inputs being coerced to an integer or double vector. 'length.out' non-negative integer. The desired length of the output vector. Other inputs will be coerced to a double vector and the first element taken. Ignored if 'NA' or invalid. 'each' non-negative integer. Each element of 'x' is repeated 'each' times. Other inputs will be coerced to an integer or double vector and the first element taken. Treated as '1' if 'NA' or invalid.

seq,nanoival-method *Sequence Generation*

Description

Generate a sequence of nanoival

Usage

```
## S4 method for signature 'nanoival'
seq(from, to = NULL, by = NULL, length.out = NULL, along.with = NULL, ...)
```

Arguments

from, to	the starting and (maximal) end values of the sequence
by	nanoduration or nanoperiod increment of the sequence; note that if the class is nanoperiod the additional argument tz must be speficied and is of character type indicating a timezone
length.out	an integer desired length of the sequence
along.with	take the length from the length of this argument.
...	arguments passed to or from methods; the only interesting additional argument is tz where the to argument is of type nanoperiod

Examples

```
## Not run:
from <- as.nanoival("-2018-01-14T13:00:00+00:00 -> 2018-01-14T15:00:00+00:00+")
seq(from, by=as.nanoperiod("1m"), length.out=5, tz="America/New_York")

## End(Not run)
```

seq.nanoduration *Sequence Generation*

Description

Generate a sequence of nanoduration

Usage

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

Arguments

from, to	the starting and (maximal) end values of the sequence
by	the increment of the sequence
length.out	integer indicating the desired length of the sequence
along.with	take the length from the length of this argument.
...	arguments passed to or from methods

Examples

```
seq(from=as.nanoduration(0), by=as.nanoduration("01:00:00"), length.out=10)
```

---

seq.nanotime *Sequence Generation*

Description

Generate a sequence of nanotime

Usage

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

## S4 method for signature 'nanotime'
seq(from, to = NULL, by = NULL, length.out = NULL, along.with = NULL, ...)
```

Arguments

from, to	the starting and (maximal) end values of the sequence
by	nanoduration or nanoperiod increment of the sequence; note that if the class is nanoperiod the additional argument tz must be speficied and is of character type indicating a timezone
length.out	integer indicating the desired length of the sequence
along.with	take the length from the length of this argument.
...	arguments passed to or from methods; the only interesting additional argument is tz where the to argument is of type nanoperiod

Examples

```
## Not run:
from <- as.nanotime("2018-01-14T12:44:00+00:00")
to   <- as.nanotime("2019-01-14T12:44:00+00:00")
seq(from, to, by=as.nanoperiod("1m"), tz="America/New_York")
seq(from, by=as.nanoperiod("1y"), length.out=4, tz="Europe/London")

## End(Not run)
```

`sort,nanoival-method` *Sorting or Ordering Vectors*

Description

Sort (or _order_) a vector of `nanoival` into ascending or descending order

Usage

```
## S4 method for signature 'nanoival'
sort(x, decreasing = FALSE)
```

Arguments

`x`	a vector of `nanoival`
`decreasing`	logical. Should the sort be increasing or decreasing?

See Also

`is.unsorted`

Index