Package 'parttime'

May 9, 2026

Type Package

Title Partial Datetime Handling

Version 0.1.2

Description Datetimes and timestamps are invariably an imprecise notation, with any partial representation implying some amount of uncertainty. To handle this, 'parttime' provides classes for embedding partial missingness as a central part of its datetime classes. This central feature allows for more ergonomic use of datetimes for challenging datetime computation, including calculations of overlapping date ranges, imputations, and more thoughtful handling of ambiguity that arises from uncertain time zones. This package was developed first and foremost with pharmaceutical applications in mind, but aims to be agnostic to application to accommodate general use cases just as conveniently.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.2.3

BugReports https://github.com/dgkf/parttime/issues

URL https://dgkf.github.io/parttime/, https://github.com/dgkf/parttime

Depends R (>= 3.6)

Imports crayon, lubridate, methods, pillar, utils, vctrs (>= 0.2.0)

Suggests dplyr, withr, testthat, knitr, rmarkdown

NeedsCompilation no

Author Doug Kelkhoff [aut, cre],
Bill Denney [ctb] (ORCID: <https://orcid.org/0000-0002-5759-428X>)

Maintainer Doug Kelkhoff <doug.kelkhoff@gmail.com>

Repository CRAN

Date/Publication 2024-01-24 23:00:03 UTC

Contents

`+,partial_time,Period-method`
Addition of a lubridate Period to a parttime partial_time

Description

Addition of a lubridate Period to a parttime partial_time

Usage

```
## S4 method for signature 'partial_time,Period'
e1 + e2
```

Arguments

`e1`	A partial_time object
`e2`	A lubridate Period object

Value

A new `partial_time` object offset by `Period` `e2`

`as.interval,partial_time-method`
Wrapper for lubridate as.interval

Description

Wrapper for lubridate as.interval

Usage

```
## S4 method for signature 'partial_time'
as.interval(x, start, ...)
```

Arguments

`x`	a duration, difftime, period, or numeric object that describes the length of the interval
`start`	a POSIXt or Date object that describes when the interval begins
`...`	additional arguments to pass to as.interval

Value

A `partial_timespan` vector

`as.interval,timespan-method`
Wrapper for lubridate as.interval

Description

Wrapper for lubridate as.interval

Usage

```
## S4 method for signature 'timespan'
as.interval(x, start, ...)
```

Arguments

`x`	a duration, difftime, period, or numeric object that describes the length of the interval
`start`	a POSIXt or Date object that describes when the interval begins
`...`	additional arguments to pass to as.interval

Value

An unaffected `partial_timespan`

`as.parttime` *Coerce an object to a parttime object*

Description

Coerce an object to a parttime object

Usage

```
as.parttime(x, ..., format = parse_iso8601_datetime, on.na = "warning")
```

Arguments

`x`	an object for coersion
`...`	Additional arguments passed to `format` when a function is provided.
`format`	a `function` or `character` value. If a `function`, it should accept a character vector and return a matrix of parttime components. If a `character` it should provide a regular exprssion which contains capture groups for each of the parttime components. See parse_to_parttime_matrix's `regex` parameter for more details.
`on.na`	a `function` used to signal a condition for new `NA` values introduced by coercion, a `character` value among `"error"`, `"warning"` or `"suppress"` (for silencing messages) or `NULL` equivalent to `"suppress"`.

Value

`parttime` vector. See the Details section of parttime for further information.

Examples

```
as.parttime(c("1985-10-18", "1991-08-23", "1996-09-26"))
# <partial_time<YMDhmsZ>[3]>
# [1] "1985-10-18" "1991-08-23" "1996-09-26"

as.parttime(c("1234", "5678"), format = "(?<year>\\d{4})")
# <partial_time<YMDhmsZ>[2]>
# [1] "1234" "5678"

# format function that returns a matrix of components
utf8_str <- function(x) intToUtf8(utf8ToInt(x) - 16)
as.parttime(c("B@", "B@A@"), format = function(x) cbind(year = sapply(x, utf8_str)))
# <partial_time<YMDhmsZ>[2]>
# [1] "2000" "2010"

# format function that returns a parttime object by first pre-processing input
as.parttime("B@BB", format = function(x) as.parttime(utf8_str(x)))
# <partial_time<YMDhmsZ>[1]>
# [1] "2022"
```

```
# format function that returns a parttime object by manual construction
as.parttime("AIII", format = function(x) parttime(year = as.numeric(utf8_str(x))))
# <partial_time<YMDhmsZ>[1]>
# [1] "1999"
```

`as.timespan` *Cast an object to a timespan*

Description

Cast an object to a timespan

Usage

```
as.timespan(x, ..., format = parse_iso8601_datetime_as_timespan)
```

Arguments

`x`	an object to cast
`...`	Additional arguments passed to `format` when a function is provided.
`format`	a `function` or `character` value. If a `function`, it should accept a character vector and return a matrix of parttime components. If a `character` it should provide a regular exprssion which contains capture groups for each of the parttime components. See parse_to_parttime_matrix's `regex` parameter for more details.

Value

A `partial_time` object. See the Details section of timespan for more information.

`c.partial_time` *Concatenate parttimes*

Description

Concatenate parttimes

Usage

```
## S3 method for class 'partial_time'
c(...)
```

Arguments

...	objects to be concatenated. All NULL entries are dropped before method dispatch unless at the very beginning of the argument list.

Value

A partial_time vector. An error is raised if any other class object is attempted to be concatenated.

---

definitely *"Definitely" generic for resolving uncertainty*

Description

"Definitely" generic for resolving uncertainty

Usage

```
definitely(x, ...)
```

Arguments

x	an uncertain object to resolve
...	additional paramters used by class-specific functions

Value

A logical vector indicating whether the partial time comparison is possibly or definitely true provided any uncertainty represented in the partial_time inputs.

See Also

Other uncert-resolvers: possibly()

---

definitely.partial_time_logical *Determine whether a partial_time logical matrix is definitely TRUE*

Description

Determine whether a partial_time logical matrix is definitely TRUE

Usage

```
## S3 method for class 'partial_time_logical'
definitely(x, by = ncol(attr(x, "pttm_lgl")), ...)
```

Arguments

x	a partial_time_logical matrix for coersion
by	the resolution of assessment, a column or index
...	additional arguments unused

Value

A logical vector indicating whether the partial time comparison is possibly or definitely true provided any uncertainty represented in the `partial_time` inputs.

Examples

```
x <- as.parttime(c("", "2019", "2018-01-02"))
y <- as.parttime(c("2018", "2019-02", "2018-02"))

definitely(x != y)
definitely(x != y, by = "year")
```

`dim.partial_time` *parttime vector dimensions*

Description

parttime vector dimensions

Usage

```
## S3 method for class 'partial_time'
dim(x)
```

Arguments

x	A `partial_time` object

Value

An `integer` vector of dimensions (length) of a `partial_time` vector

end *end S3 generic*

Description

A generic method to retrieve the end of an object

Usage

```
end(x, ...)
```

Arguments

x	An object to retrieve the end from
...	Additional arguments passed to methods

Value

The ending partial_time of a partial_timespan object.

format.partial_time *Format a parttime object*

Description

Format a parttime object

Usage

```
## S3 method for class 'partial_time'
format(x, ..., quote = TRUE)
```

Arguments

x	A partial_time object
...	Additional arguments passed to format_field_matrix
quote	A logical indicating whether to add quotation marks around formatted objects

Value

A character vector representation of a partial_time vector

format.pillar_shaft_partial_time *parttime pillar formatting*

Description

parttime pillar formatting

Usage

```
## S3 method for class 'pillar_shaft_partial_time'
format(x, width, ...)
```

Arguments

x	A character vector with formatting, can use ANYI styles e.g provided by the **cli** package.
width	A maximum display width of the each element in the resulting vector of strings
...	Additional arguments unused

Value

A character representation of a partial_time vector

has_partial *Test whether a partial_time object is incomplete*

Description

Test whether a partial_time object is incomplete

Usage

```
has_partial(
  x,
  ...,
 components = c("year", "month", "day", "hour", "min", "sec", "secfrac", "tzhour",
    "tzmin")
)
```

Arguments

x	a partial_time object to test for incompleteness
...	additional arguments unused
components	components to include in testing

Value

A logical vector indicating whether each element of a `partial_time` has any missing datetime fields.

---

has_partial_date *Test whether a partial_time object's date components are incomplete*

Description

Test whether a partial_time object's date components are incomplete

Usage

```
has_partial_date(x)
```

Arguments

`x`	a partial_time object to test for incompleteness

Value

A logical vector indicating whether each element of a `partial_time` has any missing date fields.

---

has_partial_time *Test whether a partial_time object's time components are incomplete*

Description

Test whether a partial_time object's time components are incomplete

Usage

```
has_partial_time(x)
```

Arguments

`x`	a partial_time object to test for incompleteness

Value

A logical vector indicating whether each element of a `partial_time` has any missing time fields.

impute_time *Impute a partial time object with a timestamp or specific fields*

Description

Impute a partial time object with a timestamp or specific fields

Usage

```
impute_time(x, time, tz, ...)

impute_date(x, time, ..., res = "day")

impute_time_min(x, tz = "-1200", ...)

impute_date_min(x, ..., res = "day")

impute_time_max(x, tz = "+1400", ...)

impute_date_max(x, ..., res = "day")

impute_time_mid(x, tz = "GMT", ...)

impute_date_mid(x, ..., res = "day")

## Default S3 method:
impute_time(x, time, tz = "GMT", ...)

## S3 method for class 'POSIXt'
impute_time(x, time, tz = "GMT", ...)

## S3 method for class 'partial_time'
impute_time(x, time, tz = "GMT", ..., res = NULL)
```

Arguments

x	a datetime-like object to impute
time	a datetime-like object to use for imputation
tz	a character timezone name for imputation, a character value to use as the timezone part of the datetime or an numeric minute offset.
...	additional individual named fields to impute. Can be one of "year", "month", "day", "hour", "min", "sec", "tzhour"
res	the highest resolution datetime field used for imputation. Either a character value represented the highest resolution field or NULL to impute all fields. For the impute_date family of functions, defaults to "day", or NULL otherwise.

Value

a new `partial_time` with specified fields imputed by values provided by the imputation `time`

`includes` *Determine whether one object includes another*

Description

Determine whether one object includes another

Usage

```
includes(e1, e2)
```

Arguments

`e1`	object to test whether includes `e2`
`e2`	object to test whether included in `e1`

Value

A logical vector indicating whether `e1` includes `e2`

`includes.partial_time` *Determine whether a partial time contains an object*

Description

Determine whether a partial time contains an object

Usage

```
## S3 method for class 'partial_time'
includes(e1, e2)
```

Arguments

`e1`	object to test whether includes `e2`
`e2`	object to test whether included in `e1`

Value

A logical vector indicating whether `e1` includes `e2`

includes.partial_time.partial_time

Test for whether a timestamp could be included within parttime uncertainty

Description

Test for whether a timestamp could be included within parttime uncertainty

Usage

```
## S3 method for class 'partial_time'
includes.partial_time(e1, e2)
```

Arguments

e1	object to test whether includes e2
e2	object to test whether included in e1

Value

A logical vector indicating whether partial_time e1 includes partial_time e2

Examples

```
x_chr <- c("2019", "2019-03-01", "2019-03",    "2018",    "",     "2018", "")
y_chr <- c("2019", "2019-03",    "2019-03-01", "2016-05", "2018", "",     "")

x <- as.parttime(x_chr)
y <- as.parttime(y_chr)

includes(x, y)
```

is.na.partial_time *Check if elements of a partial time vector is NA*

Description

Check if elements of a partial time vector is NA

Usage

```
## S3 method for class 'partial_time'
is.na(x, ...)
```

Arguments

x	partial_time vector to test
...	additional arguments unused

Value

A logical vector indicating whether each element in the partial_time vector is NA.

---

is.na.timespan *Check if elements of a partial time vector is NA*

Description

Check if elements of a partial time vector is NA

Usage

```
## S3 method for class 'timespan'
is.na(x, ...)
```

Arguments

x	partial_time vector to test
...	additional arguments unused

Value

A logical vector indicating whether each element in the partial_timespan vector is NA.

---

is.timespan *Shorthand for checking timespan inheritance*

Description

Shorthand for checking timespan inheritance

Usage

```
is.timespan(x)
```

Arguments

x	object to test

Value

A logical scalar indicating whether an object is a partial_timespan object.

is_partial_time *Shorthand for checking partial time inheritance*

Description

Shorthand for checking partial time inheritance

Usage

```
is_partial_time(x)

is.partial_time(x)

is_parttime(x)

is.parttime(x)
```

Arguments

x	object to test

Value

A logical scalar indicating whether an object is a `partial_time` object.

is_timespan *Shorthand for checking timespan inheritance*

Description

Shorthand for checking timespan inheritance

Usage

```
is_timespan(x)
```

Arguments

x	object to test

Value

A logical scalar indicating whether an object is a `partial_timespan` object.

max.partial_time *Get the maximum of a parttime vector*

Description

Get the maximum of a parttime vector

Usage

```
## S3 method for class 'partial_time'
max(..., na.rm = FALSE, na.warn = TRUE)
```

Arguments

...	partial_time objects
na.rm	whether NA should be removed when calculating max
na.warn	whether to raise a warning for NA

Value

A partial_time scalar

Examples

```
max(parttime(c("2019", "2018", "2019-02", "2018-03")))
```

min.partial_time *Get the minimum of aparttime vector*

Description

Get the minimum of aparttime vector

Usage

```
## S3 method for class 'partial_time'
min(..., na.rm = FALSE, na.warn = TRUE)
```

Arguments

...	partial_time objects
na.rm	whether NA should be removed when calculating min
na.warn	whether to raise a warning for NA

Value

A partial_time scalar

---

normalize_month_day *Normalize days in month back to day limit for a given month*

Description

Normalize days in month back to day limit for a given month

Usage

```
normalize_month_day(x)
```

Arguments

x	a vector of parttime objects with days which may exceed viable days of month

Value

a vector of partial_time objects with normalized days of the month

Examples

```
x <- as.parttime(c("2019", "2019-02-31", "2019-01-05", "2016-02-31", "2016-01-05"))
parttime:::normalize_month_day(x)
```

---

obj_print_data.partial_time *parttime data output*

Description

parttime data output

Usage

```
## S3 method for class 'partial_time'
obj_print_data(x, ...)
```

Arguments

x	A partial_time object
...	Additional arguments unused

Value

A `character` representation of `partial_time`

`obj_print_footer.partial_time`
parttime footer

Description

parttime footer

Usage

```
## S3 method for class 'partial_time'
obj_print_footer(x, ...)
```

Arguments

`x`	A `partial_time` object
`...`	Additional arguments unused

Value

A string output when `partial_time` vector printing exceeds max print length.

`obj_print_header.partial_time`
parttime output header

Description

parttime output header

Usage

```
## S3 method for class 'partial_time'
obj_print_header(x, ...)
```

Arguments

`x`	A `partial_time` object
`...`	Additional arguments unused

Value

A `character` representation of `partial_time` metadata, as used to describe its vector output header

`Ops.partial_time` *Handler for Ops generics for partial_time objects*

Description

Handler for Ops generics for partial_time objects

Usage

```
## S3 method for class 'partial_time'
Ops(e1, e2)
```

Arguments

`e1`	objects
`e2`	objects

Details

`partial_time` objects only implement binary operators `==` and `!=`. For other operators, `partial_times` are first converted to `partial_timespans` for operator evaluation.

Value

the binary operator result of `partial_time` `e1` with `e2`. See Details for more information on operator behaviors.

See Also

possibly definitely

Examples

```
#                 when assume_tz "GMT"         when assume_tz NA
#                 --------------------------  --------------------------
#                 raw    possibly  definitely  raw   possibly  definitely
#                 -----  --------- ----------  ----- --------- -----------
#     1998 < 1999 TRUE   TRUE      TRUE        NA    TRUE      FALSE
#     1998 < 1997 FALSE  FALSE     FALSE       NA    TRUE      FALSE
#     1999 < 1999 NA     TRUE      FALSE       NA    TRUE      FALSE
# 1998 < 1999/1/3 TRUE   TRUE      TRUE        TRUE  TRUE      TRUE

parttime(1998) < parttime(1999)
parttime(1998) < parttime(1997)
parttime(1999) < parttime(1999)
parttime(1998) < parttime(1999, 1, 3)
```

Ops.timespan *Handler for Ops generics for timespan objects*

Description

Handler for Ops generics for timespan objects

Usage

```
## S3 method for class 'timespan'
Ops(e1, e2)
```

Arguments

e1	objects
e2	objects

Value

the binary operator result of `partial_timespan` `e1` with `e2`. See Details for more information on operator behaviors.

parttime *Create a parttime object*

Description

Create a parttime object

Usage

```
parttime(
  year = NA,
  month = NA,
  day = NA,
  hour = NA,
  min = NA,
  sec = NA,
  tzhour = interpret_tz(getOption("parttime.assume_tz_offset", NA))/60
)
```

Arguments

year	numeric vector to use for partial time year component
month	numeric vector to use for partial time month component
day	numeric vector to use for partial time day component
hour	numeric vector to use for partial time hour component
min	numeric vector to use for partial time min component
sec	numeric vector to use for partial time sec component
tzhour	numeric vector to use for partial time tzhour component

Details

A parttime object (short for its class name, partial_time), is a vector representation of a numeric matrix containing rows for each vector element and a column for each datetime field.

To inspect the internal representation of a partial_time class vector, you can use vctrs::field(<pttm>, "pttm_mat").

Value

A partial_time object. See Details section for further information.

Examples

```
parttime(2019)
```

---

parttime_access_and_assign

Datetime component access and assignment functions

Description

Datetime component access and assignment functions

Usage

```
year(x)

year(x) <- value

month(x)

month(x) <- value

mday(x)

mday(x) <- value
```

```
day(x)

day(x) <- value

hour(x)

hour(x) <- value

minute(x)

minute(x) <- value

second(x)

second(x) <- value

tz(x)

tz(x) <- value

## S3 method for class 'partial_time'
year(x)

## S3 replacement method for class 'partial_time'
year(x) <- value

## S4 replacement method for signature 'partial_time'
year(x) <- value

## S3 method for class 'partial_time'
month(x)

## S3 replacement method for class 'partial_time'
month(x) <- value

## S4 replacement method for signature 'partial_time'
month(x) <- value

## S3 method for class 'partial_time'
mday(x)

## S3 replacement method for class 'partial_time'
day(x) <- value

## S4 replacement method for signature 'partial_time'
day(x) <- value
```

```
## S3 method for class 'partial_time'
hour(x)

## S3 replacement method for class 'partial_time'
hour(x) <- value

## S4 replacement method for signature 'partial_time'
hour(x) <- value

## S3 method for class 'partial_time'
minute(x)

## S3 replacement method for class 'partial_time'
minute(x) <- value

## S4 replacement method for signature 'partial_time'
minute(x) <- value

## S3 method for class 'partial_time'
second(x)

## S3 replacement method for class 'partial_time'
second(x) <- value

## S4 replacement method for signature 'partial_time'
second(x) <- value

## S3 method for class 'partial_time'
tz(x)

## S3 replacement method for class 'partial_time'
tz(x) <- value
```

Arguments

x	A time-like object to access or assign to
value	For assignment, a value to assign

Value

The numeric vector associated with the accessor field.

Note

Care is taken to make these functionas as compatible as possible with similar datetime packages. However, some functions may be masked and cause errors using their masking functions.

parttime_extract *Indexing operators for partial_time objects*

Description

Indexing operators repurpose matrix indexing for indexing into parttime fields. When only `i` is provided, the `parttime` vector is sliced. Whenever `j` is provided, the individual fields are indexed out of an internal matrix.

Usage

```
## S3 method for class 'partial_time'
x[i, j, ...]

## S3 method for class 'partial_time'
x[[i, j, ..., value]]

## S3 replacement method for class 'partial_time'
x[i, j, ..., reflow = TRUE] <- value

## S3 replacement method for class 'partial_time'
x[[i, ...]] <- value
```

Arguments

`x`	an object from which to extract element(s) or in which to replace element(s).
`i`	indicies specifying elements to extract or replace. For further details, see Extract.
`j`	column indicies specifying element(s) to extract or replace. For further details, see Extract.
`...`	arguments unused
`value`	typically an array-like R object of a similar class as `x`.
`reflow`	a `logical` indicating whether modified data fields should be reflowed, cascading range overflow. Setting to `FALSE` permits invalid dates, but saves on compute. Generally, it should only be disabled when multiple calculations are performed back-to-back and the dates only need to be reflowed once at the end of the calculation.

Value

A numeric matrix subset of the `partial_time` internal matrix representation. See the Details section of parttime for further information.

A `numeric` vector of the provided parttime field

the new value of the assigned `partial_time` object after modification.

A `partial_time` vector after modification

Examples

```
x <- as.parttime(c("2019", "2019-02", "2019-02-02"))
# <partial_time<YMDhms+tz>[3]>
# [1] "2019"       "2019-02"    "2019-02-02"

x[, c(1, 3)]
#            year day
# 2019       2019  NA
# 2019-02    2019  NA
# 2019-02-02 2019   2

x[, "month"]
#       2019    2019-02 2019-02-02
#         NA          2          2

x[, "month", drop = FALSE]
#            month
# 2019          NA
# 2019-02        2
# 2019-02-02     2

x <- as.parttime(c("2019", "2019-02", "2019-02-02"))
# <partial_time<YMDhms+tz>[3]>
# [1] "2019"       "2019-02"    "2019-02-02"

x[c(1, 3)] <- as.parttime(c("2000", "1999"))
# <partial_time<YMDhms+tz>[3]>
# [1] "2000"    "2019-02" "1999"

x[,"month"] <- 3
# <partial_time<YMDhms+tz>[3]>
# [1] "2000-03" "2019-03" "1999-03"
```

---

pillar_shaft.partial_time

parttime as pillar shaft

Description

parttime as pillar shaft

Usage

```
## S3 method for class 'partial_time'
pillar_shaft(x, ...)
```

Arguments

x	An object
...	Passed on to new_pillar_shaft().

Value

A character representation of a partial_time vector

---

pmax *Maxima and Minima*

Description

Returns the (regular or **p**arallel) maxima and minima of the input values.

pmax*() and pmin*() take one or more vectors as arguments, recycle them to common length and return a single vector giving the *'parallel'* maxima (or minima) of the argument vectors.

Usage

```
pmax(..., na.rm = FALSE)
```

Arguments

...	numeric or character arguments (see Note).
na.rm	a logical indicating whether missing values should be removed.

Details

max and min return the maximum or minimum of *all* the values present in their arguments, as integer if all are logical or integer, as double if all are numeric, and character otherwise.

If na.rm is FALSE an NA value in any of the arguments will cause a value of NA to be returned, otherwise NA values are ignored.

The minimum and maximum of a numeric empty set are +Inf and -Inf (in this order!) which ensures *transitivity*, e.g., min(x1, min(x2)) == min(x1, x2). For numeric x max(x) == -Inf and min(x) == +Inf whenever length(x) == 0 (after removing missing values if requested). However, pmax and pmin return NA if all the parallel elements are NA even for na.rm = TRUE.

pmax and pmin take one or more vectors (or matrices) as arguments and return a single vector giving the 'parallel' maxima (or minima) of the vectors. The first element of the result is the maximum (minimum) of the first elements of all the arguments, the second element of the result is the maximum (minimum) of the second elements of all the arguments and so on. Shorter inputs (of non-zero length) are recycled if necessary. Attributes (see attributes: such as names or dim) are copied from the first argument (if applicable, e.g., *not* for an S4 object).

pmax.int and pmin.int are faster internal versions only used when all arguments are atomic vectors and there are no classes: they drop all attributes. (Note that all versions fail for raw and complex vectors since these have no ordering.)

max and min are generic functions: methods can be defined for them individually or via the Summary group generic. For this to work properly, the arguments ... should be unnamed, and dispatch is on the first argument.

By definition the min/max of a numeric vector containing an NaN is NaN, except that the min/max of any vector containing an NA is NA even if it also contains an NaN. Note that max(NA, Inf) == NA even though the maximum would be Inf whatever the missing value actually is.

Character versions are sorted lexicographically, and this depends on the collating sequence of the locale in use: the help for 'Comparison' gives details. The max/min of an empty character vector is defined to be character NA. (One could argue that as "" is the smallest character element, the maximum should be "", but there is no obvious candidate for the minimum.)

Value

For min or max, a length-one vector. For pmin or pmax, a vector of length the longest of the input vectors, or length zero if one of the inputs had zero length.

The type of the result will be that of the highest of the inputs in the hierarchy integer < double < character.

For min and max if there are only numeric inputs and all are empty (after possible removal of NAs), the result is double (Inf or -Inf).

S4 methods

max and min are part of the S4 Summary group generic. Methods for them must use the signature x, ..., na.rm.

Note

'Numeric' arguments are vectors of type integer and numeric, and logical (coerced to integer). For historical reasons, NULL is accepted as equivalent to integer(0).

pmax and pmin will also work on classed S3 or S4 objects with appropriate methods for comparison, is.na and rep (if recycling of arguments is needed).

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

See Also

range (*both* min and max) and which.min (which.max) for the *arg min*, i.e., the location where an extreme value occurs.

'plotmath' for the use of min in plot annotation.

Examples

```
require(stats); require(graphics)
 min(5:1, pi) #-> one number
pmin(5:1, pi) #->  5  numbers
```

```
x <- sort(rnorm(100));  cH <- 1.35
pmin(cH, quantile(x)) # no names
pmin(quantile(x), cH) # has names
plot(x, pmin(cH, pmax(-cH, x)), type = "b", main =  "Huber's function")

cut01 <- function(x) pmax(pmin(x, 1), 0)
curve(      x^2 - 1/4, -1.4, 1.5, col = 2)
curve(cut01(x^2 - 1/4), col = "blue", add = TRUE, n = 500)
## pmax(), pmin() preserve attributes of *first* argument
D <- diag(x = (3:1)/4) ; n0 <- numeric()
stopifnot(identical(D,  cut01(D) ),
          identical(n0, cut01(n0)),
          identical(n0, cut01(NULL)),
          identical(n0, pmax(3:1, n0, 2)),
          identical(n0, pmax(n0, 4)))
```

`pmax.partial_time` *Get the elementwise maximum of parttime vectors*

Description

Get the elementwise maximum of parttime vectors

Usage

```
## S3 method for class 'partial_time'
pmax(..., na.rm = FALSE)
```

Arguments

`...`	numeric or character arguments (see Note).
`na.rm`	a logical indicating whether missing values should be removed.

Value

A `partial_time` vector with length equal to the maximum length of the vectors provided where each value is the maximum of the recycled values of each vector argument.

Examples

```
pmax(
  parttime(c("2019", "2018", "2019-02", "2018",    "2010")),
  parttime(c("2020", NA,     "2019-03", "2018-01", "2010"))
)
```

pmin *Maxima and Minima*

Description

Returns the (regular or **p**arallel) maxima and minima of the input values.

`pmax*()` and `pmin*()` take one or more vectors as arguments, recycle them to common length and return a single vector giving the *'parallel'* maxima (or minima) of the argument vectors.

Usage

```
pmin(..., na.rm = FALSE)
```

Arguments

`...`	numeric or character arguments (see Note).
`na.rm`	a logical indicating whether missing values should be removed.

Details

`max` and `min` return the maximum or minimum of *all* the values present in their arguments, as `integer` if all are `logical` or `integer`, as `double` if all are numeric, and character otherwise.

If `na.rm` is `FALSE` an `NA` value in any of the arguments will cause a value of `NA` to be returned, otherwise `NA` values are ignored.

The minimum and maximum of a numeric empty set are `+Inf` and `-Inf` (in this order!) which ensures *transitivity*, e.g., `min(x1, min(x2)) == min(x1, x2)`. For numeric `x` `max(x) == -Inf` and `min(x) == +Inf` whenever `length(x) == 0` (after removing missing values if requested). However, `pmax` and `pmin` return `NA` if all the parallel elements are `NA` even for `na.rm = TRUE`.

`pmax` and `pmin` take one or more vectors (or matrices) as arguments and return a single vector giving the 'parallel' maxima (or minima) of the vectors. The first element of the result is the maximum (minimum) of the first elements of all the arguments, the second element of the result is the maximum (minimum) of the second elements of all the arguments and so on. Shorter inputs (of non-zero length) are recycled if necessary. Attributes (see `attributes`: such as `names` or `dim`) are copied from the first argument (if applicable, e.g., *not* for an `S4` object).

`pmax.int` and `pmin.int` are faster internal versions only used when all arguments are atomic vectors and there are no classes: they drop all attributes. (Note that all versions fail for raw and complex vectors since these have no ordering.)

`max` and `min` are generic functions: methods can be defined for them individually or via the `Summary` group generic. For this to work properly, the arguments `...` should be unnamed, and dispatch is on the first argument.

By definition the min/max of a numeric vector containing an `NaN` is `NaN`, except that the min/max of any vector containing an `NA` is `NA` even if it also contains an `NaN`. Note that `max(NA, Inf) == NA` even though the maximum would be `Inf` whatever the missing value actually is.

Character versions are sorted lexicographically, and this depends on the collating sequence of the locale in use: the help for 'Comparison' gives details. The max/min of an empty character vector

is defined to be character NA. (One could argue that as "" is the smallest character element, the maximum should be "", but there is no obvious candidate for the minimum.)

Value

For min or max, a length-one vector. For pmin or pmax, a vector of length the longest of the input vectors, or length zero if one of the inputs had zero length.

The type of the result will be that of the highest of the inputs in the hierarchy integer < double < character.

For min and max if there are only numeric inputs and all are empty (after possible removal of NAs), the result is double (Inf or -Inf).

S4 methods

max and min are part of the S4 Summary group generic. Methods for them must use the signature x, ..., na.rm.

Note

'Numeric' arguments are vectors of type integer and numeric, and logical (coerced to integer). For historical reasons, NULL is accepted as equivalent to integer(0).

pmax and pmin will also work on classed S3 or S4 objects with appropriate methods for comparison, is.na and rep (if recycling of arguments is needed).

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

See Also

range (*both* min and max) and which.min (which.max) for the *arg min*, i.e., the location where an extreme value occurs.

'plotmath' for the use of min in plot annotation.

Examples

```
require(stats); require(graphics)
 min(5:1, pi) #-> one number
pmin(5:1, pi) #->  5  numbers

x <- sort(rnorm(100));  cH <- 1.35
pmin(cH, quantile(x)) # no names
pmin(quantile(x), cH) # has names
plot(x, pmin(cH, pmax(-cH, x)), type = "b", main =  "Huber's function")

cut01 <- function(x) pmax(pmin(x, 1), 0)
curve(      x^2 - 1/4, -1.4, 1.5, col = 2)
curve(cut01(x^2 - 1/4), col = "blue", add = TRUE, n = 500)
## pmax(), pmin() preserve attributes of *first* argument
```

```
D <- diag(x = (3:1)/4) ; n0 <- numeric()
stopifnot(identical(D,  cut01(D) ),
          identical(n0, cut01(n0)),
          identical(n0, cut01(NULL)),
          identical(n0, pmax(3:1, n0, 2)),
          identical(n0, pmax(n0, 4)))
```

`pmin.partial_time` *Get the elementwise minimum of parttime vectors*

Description

Get the elementwise minimum of parttime vectors

Usage

```
## S3 method for class 'partial_time'
pmin(..., na.rm = FALSE)
```

Arguments

`...`	numeric or character arguments (see Note).
`na.rm`	a logical indicating whether missing values should be removed.

Value

A `partial_time` vector with length equal to the maximum length of the vectors provided where each value is the minimum of the recycled values of each vector argument.

`possibly` *"Possibly" generic for resolving uncertainty*

Description

"Possibly" generic for resolving uncertainty

Usage

```
possibly(x, ...)
```

Arguments

`x`	an uncertain object to resolve
`...`	additional paramters used by class-specific functions

Value

A logical vector indicating whether the partial time comparison is possibly or definitely true provided any uncertainty represented in the `partial_time` inputs.

See Also

Other uncert-resolvers: `definitely()`

`possibly.partial_time_logical` *Determine whether a partial_time logical matrix is possibly TRUE*

Description

Determine whether a partial_time logical matrix is possibly TRUE

Usage

```
## S3 method for class 'partial_time_logical'
possibly(x, by = ncol(attr(x, "pttm_lgl")), ...)
```

Arguments

`x`	a partial_time_logical matrix for coersion
`by`	the resolution of assessment, a column or index
`...`	additional arguments unused

Value

A logical vector indicating whether the partial time comparison is possibly or definitely true provided any uncertainty represented in the `partial_time` inputs.

Examples

```
x <- as.parttime(c("",     "2019-02",    "2019-01-02"))
y <- as.parttime(c("2018", "2019-02-01", "2018"))

possibly(x != y)
possibly(x != y, by = "month")
```

`reflow_fields` *Reflow potentially invalid time components to adjacent fields*

Description

Reflow potentially invalid time components to adjacent fields

Usage

```
reflow_fields(fmat, days)
```

Arguments

`fmat`	a fields matrix as part of a partial_time or partial_difftime
`days`	a logical indicating whether year and month should be consolidated into total days. If an integer is provided, days should represent the "leap-time" to add on top of non-leap conversion.

Value

a fields matrix with appropriately ranged time components

Examples

```
# example with difftimes (when you only care about days of change)

x <- as.parttime("2019-06-23 04:33:21.123")
y <- as.parttime("2018-02-08 12:59:28.987")

diff_fields <- vctrs::field(x, "pttm_mat") - vctrs::field(y, "pttm_mat")

parttime:::reflow_fields(diff_fields)

# if we want to assume 0.25 leap days per year
parttime:::reflow_fields(diff_fields, days = TRUE)

# if we want to assert that there were no leap days
parttime:::reflow_fields(diff_fields, days = 0)
```

start *start S3 generic*

Description

A generic method to retrieve the start of an object

Usage

```
start(x, ...)
```

Arguments

x	An object to retrieve the start from
...	Additional arguments passed to methods

Value

The starting partial_time of a partial_timespan object.

timespan *Create a partial timespan object*

Description

Create a partial timespan object

Usage

```
timespan(start, end, inclusive = c(TRUE, FALSE))
```

Arguments

start	vector of datetime objects to start timespans
end	vector of datetime objects to end timespans
inclusive	vector or matrix of logicals where each row is composed of two logical values indicating whether the timespan start and end are inclusive respectively

Details

Partial timespans are vector representations of an array of (possibly missing) datetime fields. They represent timespans while accounting for the possibility that their start and end might not be fully known. The start and end are represented similarly to partial_time objects, and represent a lower and upper bound for the timespan, and may be either inclusive or exclusive.

Internally, partial_timespan objects are represented as a three-dimensional array of partial time fields, with an added column representing whether each time is inclusive or exclusive. You may inspect this representation using vctrs::field(<tmspn>, "tmspn_arr").

Value

A `partial_timespan` object. See Details for further information.

`to_gmt` *Generic for coercing timestamps to GMT timezone*

Description

Generic for coercing timestamps to GMT timezone

Usage

```
to_gmt(x)
```

Arguments

`x`	object to coerce to GMT time

Value

A time object adjusted to GMT time

`trim` *Shorten a timespan*

Description

Shorten a timespan

Usage

```
trim(x, ...)
```

Arguments

`x`	timespan object to trim
`...`	additional arguments passed on to functions

type_sum.partial_time *parttime type name*

Description

parttime type name

Usage

```
## S3 method for class 'partial_time'
type_sum(x)
```

Arguments

x	an object to summarise. Generally only methods of atomic vectors and variants have been implemented.

Value

A character scalar shorthand representation of the partial_time class name

vec_cast.logical.partial_time *Cast partial time to logical*

Description

Cast partial time to logical

Usage

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

Arguments

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

Value

A partial_time vector

vec_cast.partial_time *Cast to partial time object*

Description

Cast to partial time object

Usage

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

Arguments

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

Value

A partial_time vector

vec_cast.partial_time.character *Coerce character date representations to parttime objects*

Description

Coerce character date representations to parttime objects

Usage

```
## S3 method for class 'partial_time.character'
vec_cast(x, to, ..., format = parse_iso8601_datetime, on.na = warning)
```

Arguments

x	Vectors to cast.
to	Type to cast to. If NULL, x will be returned as is.
...	Additional arguments passed to format if a function is provided.

format — a function or character value. If a function, it should accept a character vector and return a matrix of parttime components. If a character it should provide a regular exprssion which contains capture groups for each of the parttime components. See parse_to_parttime_matrix's regex parameter for more details.

on.na — a function used to signal a condition for new NA values introduced by coercion, a character value among "error", "warning" or "suppress" (for silencing messages) or NULL equivalent to "suppress".

Value

A partial_time vector

Examples

```
dates <- c(
  NA,
  "2001",
  "2002-01-01",
  "2004-245", # yearday
  "2005-W13", # yearweek
  "2006-W02-5", # yearweek + weekday
  "2007-10-01T08",
  "2008-09-20T08:35",
  "2009-08-12T08:35.048", # fractional minute
  "2010-07-22T08:35:32",
  "2011-06-13T08:35:32.123", # fractional second
  "2012-05-23T08:35:32.123Z", # Zulu time
  "2013-04-14T08:35:32.123+05", # time offset from GMT
  "2014-03-24T08:35:32.123+05:30", # time offset with min from GMT
  "20150101T083532.123+0530" # condensed form
)

as.parttime(dates)
```

---

vec_cast.partial_time.default

Default handler for casting to a partial time

Description

Default handler for casting to a partial time

Usage

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

Arguments

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

Value

A partial_time vector

---

vec_cast.partial_time.matrix

Cast a matrix to a partial time

Description

Cast a matrix to a partial time

Usage

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

Arguments

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

Value

A partial_time vector

vec_cast.timespan *Cast to timespan object*

Description

Cast to timespan object

Usage

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

Arguments

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

Value

A partial_timespan vector

vec_cast.timespan.character *Cast partial time to timespan, representing uncertainty as a range*

Description

Cast partial time to timespan, representing uncertainty as a range

Usage

```
## S3 method for class 'timespan.character'
vec_cast(x, to, ..., format = parse_iso8601_datetime_as_timespan)
```

Arguments

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

format	a function or character value. If a function, it should accept a character vector and return a matrix of parttime components. If a character it should provide a regular exprssion which contains capture groups for each of the parttime components. See parse_to_parttime_matrix's regex parameter for more details.

Value

A partial_timespan vector

---

vec_cast.timespan.default

Default handler for casting to a timespan

Description

Default handler for casting to a timespan

Usage

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

Arguments

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

Value

A partial_timespan vector

vec_cast.timespan.double

Cast an array to a timespan

Description

Cast an array to a timespan

Usage

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

Arguments

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

Value

A partial_timespan vector

vec_cast.timespan.numeric

Cast an array to a timespan

Description

Cast an array to a timespan

Usage

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

Arguments

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

Value

A partial_timespan vector

---

vec_cast.timespan.partial_time

Cast partial time to timespan, representing uncertainty as a range

---

Description

Cast partial time to timespan, representing uncertainty as a range

Usage

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

Arguments

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

Value

A partial_timespan vector

---

vec_ptype_abbr.partial_time

Abbreviated partial time class name

Description

Abbreviated partial time class name

Usage

```
## S3 method for class 'partial_time'
vec_ptype_abbr(x, ..., prefix_named, suffix_shape)
```

Arguments

x	A partial_time object
...	These dots are for future extensions and must be empty.
prefix_named	If TRUE, add a prefix for named vectors.
suffix_shape	If TRUE (the default), append the shape of the vector.

Value

A character representation of the abbreviated partial_time class name

---

vec_ptype_full.partial_time
Full parttime class name

Description

Full parttime class name

Usage

```
## S3 method for class 'partial_time'
vec_ptype_full(x, ...)
```

Arguments

x	A partial_time object
...	These dots are for future extensions and must be empty.

Value

A character representation of the partial_time class name

Index