Package ‘hms’

January 7, 2020

Title  Pretty Time of Day
Date   2020-01-07
Version 0.5.3
Description  Implements an S3 class for storing and formatting time-of-day values, based on the ‘difftime’ class.
Imports  methods,
         pkgconfig,
         rlang,
         vctrs (>= 0.2.1)
Suggests  crayon,
          lubridate,
          pillar (>= 1.1.0),
          testthat
License  GPL-3
Encoding UTF-8
LazyData true
BugReports https://github.com/tidyverse/hms/issues
RoxygenNote 7.0.2
Roxygen list(markdown = TRUE)

R topics documented:

  hms-package .................................................. 2
  hms .......................................................... 2
  parse_hms .................................................... 4
  round_hms .................................................... 5
  vec_cast_hms ................................................ 5
  vec_ptype2_hms .............................................. 6

Index 7
Description

Implements an S3 class for storing and formatting time-of-day values, based on the 'difftime' class.

Details

Stable

Author(s)

Maintainer: Kirill Müller <krlmlr+r@mailbox.org>
Other contributors:

• The R Consortium [funder]
• RStudio [funder]

See Also

Useful links:

• https://hms.tidyverse.org/
• https://github.com/tidyverse/hms
• Report bugs at https://github.com/tidyverse/hms/issues

Description

The values are stored as a *difftime* vector with a custom class, and always with "seconds" as unit for robust coercion to numeric. Supports construction from time values, coercion to and from various data types, and formatting. Can be used as a regular column in a data frame.

*hms() is a high-level constructor that accepts second, minute, hour and day components as numeric vectors.*

*new_hms() is a low-level constructor that only checks that its input has the correct base type, numeric.*

*is_hms() checks if an object is of class hms.*

*as_hms() forwards to vec_cast().*
hms

Usage

hms(seconds = NULL, minutes = NULL, hours = NULL, days = NULL)

new_hms(x = numeric())

is_hms(x)

as_hms(x)

## S3 method for class 'hms'
as.POSIXct(x, ...)

## S3 method for class 'hms'
as.POSIXlt(x, ...)

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

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

## S3 method for class 'hms'
print(x, ...)

Arguments

seconds, minutes, hours, days

Time since midnight. No bounds checking is performed.

x

An object.

... additional arguments to be passed to or from methods.

Details

For hms, all arguments must have the same length or be NULL. Odd combinations (e.g., passing only seconds and hours but not minutes) are rejected.

For arguments of type POSIXct and POSIXlt, as_hms() does not perform timezone conversion. Use lubridate::with_tz() and lubridate::force_tz() as necessary.

Examples

hms(56, 34, 12)

hms()

new_hms(as.numeric(1:3))

# Supports numeric only!

try(new_hms(1:3))

as_hms(1)
```
as_hms("12:34:56")
as_hms(Sys.time())
as.POSIXct(hms(1))
data.frame(a = hms(1))
d <- data.frame(hours = 1:3)
d$hours <- hms(hours = d$hours)
d
```

---

**parse_hms**  
*Parsing hms values*

**Description**
These functions convert character vectors to objects of the `hms` class. NA values are supported. `parse_hms()` accepts values of the form "HH:MM:SS", with optional fractional seconds. `parse_hm()` accepts values of the form "HH:MM".

**Usage**
```
parse_hms(x)
parse_hm(x)
```

**Arguments**
- `x`  
  A character vector

**Value**
An object of class `hms`.

**Examples**
```
parse_hms("12:34:56")
parse_hms("12:34:56.789")
parse_hm("12:34")
```
**round_hms**

**Round or truncate to a multiple of seconds**

**Description**
Convenience functions to round or truncate to a multiple of seconds.

**Usage**

```r
round_hms(x, secs)
trunc_hms(x, secs)
```

**Arguments**

- `x`: A vector of class `hms`
- `secs`: Multiple of seconds, a positive numeric. Values less than one are supported.

**Value**

The input, rounded or truncated to the nearest multiple of `secs`.

**Examples**

```r
cround_hms(as_hms("12:34:56"), 5)
cround_hms(as_hms("12:34:56"), 60)
ctrunc_hms(as_hms("12:34:56"), 60)
```

---

**vec_cast.hms**

**Casting**

**Description**
Double dispatch methods to support `vctrs::vec_cast()`.

**Usage**

```r
## S3 method for class 'hms'
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()` and `vec_restore()`, these dots are only for future extensions and should be empty.
Description

Double dispatch methods to support \texttt{vctrs::vec\_ptype2()}. 

Usage

```r
## S3 method for class 'hms'
vec\_ptype2(x, y, ..., x\_arg = "", y\_arg = ")
```

Arguments

- \texttt{x} Vector types.
- \texttt{y} Vector types.
- \texttt{...} These dots are for future extensions and must be empty.
- \texttt{x\_arg} Argument names for \texttt{x} and \texttt{y}. These are used in error messages to inform the user about the locations of incompatible types (see \texttt{stop\_incompatible\_type()}).
- \texttt{y\_arg} Argument names for \texttt{x} and \texttt{y}. These are used in error messages to inform the user about the locations of incompatible types (see \texttt{stop\_incompatible\_type()}).
Index

as.character.hms (hms), 2
as.POSIXct.hms (hms), 2
as.POSIXlt.hms (hms), 2
as_hms (hms), 2
difftime, 2
format.hms (hms), 2
hms, 2, 4, 5
hms-package, 2
is_hms (hms), 2
lubridate::force_tz(), 3
lubridate::with_tz(), 3
new_hms (hms), 2
numeric, 2
parse.hm (parse_hms), 4
parse_hms, 4
POSIXct, 3
POSIXlt, 3
print.hms (hms), 2
round_hms, 5
stop_incompatible_type(), 6
trunc.hms (round_hms), 5
vctrs::vec_cast(), 5
vctrs::vec_ptype2(), 6
vec_cast(), 2
vec_cast.hms, 5
vec_ptype2.hms, 6